# TICKET RESERVATION SYSTEM


**Submitted By**

**Md Redwanul Haque**

**172-35-2116**

**Supervised By**

**Ms. Masrufa Tasnim**

**Lecturer**

**Department of Software Engineering**

**Faculty of Science and Information Technology**

**Daffodil International University**


This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

## APPROVAL

This thesis titled on "**Ticket Reservation System**", submitted by **Md Redwanul Haque (ID: 172-35-2116)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

## BOARD OF EXAMINERS

---

**Professor Dr. Engr. AKM Masum**
**Professor**
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

**Chairman**

---

**Md. Shohel Arman**
**Assistant Professor**
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

**Internal Examiner 1**

---

**Dr. Marzia Ahmed**
**Assistant**
**Professor**
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

**Internal Examiner 2**

---

**Dr. Md. Monowarul Islam**
**Associate Professor**
Department of Computer Science & Engineering
Jagannath University

**External Examiner**

## SUPERVISOR's DECLARATION

I hereby declare that I have checked this project and in my opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.

(Supervisor's Signature)

Full Name : Ms. Masrufa Tasnim
Position : Lecturer
Date : 14/01/2025

## STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

Redwanul

**(Student's Signature)**

Full Name : **Md Redwanul Haque**
ID Number : 172-35-2116
Date : 14/01/2025

# ACKNOWLEDGEMENTS

We begin by expressing our profound gratitude to The Almighty Allah, whose blessings and guidance have enabled us to successfully complete this thesis. We extend our heartfelt thanks to our supervisor, Masrufa Tasnim Esha, from the Department of Software Engineering, for her unwavering support, insightful advice, and invaluable contributions throughout this journey.

Our appreciation also goes to the faculty members of the Department of Software Engineering for their continuous support and encouragement, which motivated us to achieve our goals.

Lastly, we are profoundly grateful to our parents for their unconditional love, sacrifices, and unwavering belief in us. Their encouragement has been a pillar of strength throughout this endeavor.

# DEDICATION

I hereby dedicate this project to my respected supervisor, Masrufa Tasnim Esha, Lecturer Department of Software Engineering, Daffodil International University, for her unwavering guidance and encouragement throughout the journey of this work. I also declare that this work is my original effort and has not been submitted in full or in part to any other institution for the fulfillment of any degree or certification.

# ABSTRACT

This project, titled "Ticket Reservation System," aims to develop an efficient and user-friendly mobile application for booking tickets for events and transportation. Leveraging the Flutter framework, the application provides functionalities such as ticket browsing, booking, offline access, and quick rebooking. The system addresses gaps in existing ticket reservation platforms by simplifying processes, improving accessibility, and enhancing the overall user experience. Designed with general consumers, frequent travelers, and event organizers in mind, the project emphasizes usability, reliability, and performance within the constraints of a mobile-first solution. Through this endeavor, we aim to create a seamless and efficient ticketing solution that caters to diverse user needs.

# TABLE OF CONTENT



## LIST OF TABLES



## LIST OF FIGURES



## LIST OF APPENDICES



# CHAPTER 1 INTRODUCTION

## 1.1 Background

The rapid growth of technology has transformed the way people interact with ticketing systems for events and transportation. Despite the advancements, many existing platforms remain cumbersome, requiring users to navigate through multiple steps to book and manage tickets. Furthermore, the lack of offline accessibility and quick rebooking features often leads to inefficiencies and dissatisfaction among users.

Recognizing these challenges, this project introduces a streamlined Ticket Reservation System designed to provide a seamless booking experience through a mobile application built using Flutter. The application focuses on addressing key pain points such as complex booking processes, limited offline functionality, and poor user interface designs. By offering features like offline ticket access, intuitive navigation, and quick rebooking, this system aims to enhance user convenience and efficiency.

Targeting a diverse user base, including general consumers, frequent travelers, and event organizers, the project prioritizes simplicity, usability, and core functionalities over advanced integrations. By leveraging reliable technologies such as SQLite and Firebase, this project aspires to create a robust and practical solution that meets the needs of modern ticketing systems while ensuring accessibility and ease of use for all stakeholders.

### 1.1.1 Context and Relevance

The ticket reservation industry is a vital segment of the transportation and event management domains, playing a critical role in connecting users with services. With the increasing adoption of digital platforms, mobile-based ticketing solutions have become indispensable, offering convenience and flexibility. However, these advancements bring challenges such as maintaining user trust, ensuring data security, and meeting user expectations for seamless functionality.

Current trends in this industry emphasize personalization, mobile-first designs, and enhanced accessibility, enabling users to manage their reservations efficiently. Despite

these advancements, many systems fail to cater to users in low-connectivity environments or provide streamlined rebooking capabilities. This gap highlights a significant opportunity to innovate by addressing these unmet needs. By focusing on an intuitive user interface, offline functionality, and simplified workflows, this project aligns with the broader goal of improving user experiences in digital ticketing solutions while staying relevant to contemporary technological advancements and industry demands.

### 1.1.2 Problem Identification

Existing ticket reservation systems often struggle with inefficiencies that affect user satisfaction. These issues include overly complicated booking processes, limited offline functionality, and inadequate support for quick rebooking. While many platforms focus on offering a range of features, they frequently neglect the user experience, leading to frustration and lower adoption rates.

Prior studies and systems in the ticket reservation domain have attempted to address these challenges but have fallen short in specific areas. For example, some systems rely entirely on internet connectivity, rendering them inaccessible in areas with poor network coverage. Others lack intuitive interfaces, making navigation cumbersome for users unfamiliar with technology. Additionally, advanced features such as rebooking or integration with personal schedules are often absent or overly complex.

This project aims to bridge these gaps by creating a mobile application that prioritizes simplicity, usability, and accessibility. By integrating offline ticket access, streamlined booking workflows, and rebooking features, the proposed system addresses the core pain points identified in existing solutions. Leveraging the Flutter framework and reliable technologies such as SQLite and Firebase ensures a robust and user-centric approach to solving these problems.

### 1.1.3 Purpose and Justification

The primary purpose of this project is to enhance the ticket booking experience by creating a mobile application that is intuitive, efficient, and accessible. Current ticketing systems often fail to address key user needs, such as offline access and simplified rebooking, resulting in a fragmented and unsatisfactory experience for users. This project seeks to overcome these limitations by prioritizing core functionalities that directly address user pain points.

The justification for this project lies in its ability to provide value to multiple stakeholders, including general consumers, frequent travelers, and event organizers. By offering offline ticket access, quick rebooking options, and an intuitive user interface, the application ensures greater user satisfaction and convenience. For businesses and organizers, the system provides an effective platform for managing ticket sales and improving customer engagement.

Furthermore, the use of Flutter ensures cross-platform compatibility, allowing the application to reach a wider audience without additional development overhead. By leveraging SQLite for offline functionality and Firebase for cloud synchronization, the system combines reliability and efficiency, making it a practical solution for modern ticketing needs. This project not only fills the gaps in existing systems but also aligns with current trends in user-centered design and mobile-first development, ensuring its relevance and long-term impact.

### 1.1.4 Scope

The scope of this project encompasses the development of a mobile application that simplifies the ticket booking process for events and transportation. The primary features include:

1. **Ticket Browsing:** Users can browse and filter tickets based on events, locations, or dates.
2. **Booking Process:** A streamlined booking workflow ensures users can reserve tickets with minimal steps.

3. **Offline Access:** Booked tickets are accessible offline, enabling users to retrieve their tickets without internet connectivity.
4. **Quick Rebooking:** Users can easily rebook previously purchased tickets for recurring events or routes.
5. **Booking History:** A comprehensive history of past and current bookings is provided for user reference.
6. **Calendar Sync:** Ticket reservations can be synced with users' mobile calendars for event reminders.

The project is limited to mobile platforms and will use technologies such as Flutter, SQLite, and Firebase to ensure a robust and reliable system. Advanced features like artificial intelligence, blockchain, or complex integrations are excluded due to time and resource constraints. The focus remains on delivering a practical and user-friendly solution that meets the core needs of target users within the defined timeline.

## 1.2 Project Planning and Initiation

**Feasibility Study (Step-by-Step)**

1. **Preliminary Analysis & Project Scope Definition:**
    - Evaluate the technology stack, including Flutter, SQLite, and Firebase, to ensure they meet project requirements.
    - Assess the team's technical skills and experience in mobile application development.
2. **Market Feasibility:** The estimated budget for this project includes:
    - **Flutter SDK and Tools:** Free (open source).
    - **Firebase Usage:** $25/month for small-scale application hosting during the testing phase.
    - **SQLite Database:** Free (open source).
    - **Hosting Services for Backend Data:** $10/month.
    - **Testing Devices:** $300 for acquiring low-cost Android and iOS devices for testing.
    - **Developer Salary (Part-Time):** $1,500 for 10 weeks.
    - **Designer (UI/UX, Part-Time):** $500.

    - **Tester (Manual, Part-Time):** $300.
    - **Miscellaneous Costs (Internet, Licenses, Contingency):** $500.
3. **Total Estimated Cost:** $3,185.
4. **Operational Feasibility:**
    - Ensure that the system design aligns with user expectations for simplicity and usability.
    - Plan for regular updates and maintenance to meet evolving user needs.
5. **Legal Feasibility:**
    - Verify compliance with data protection regulations, including GDPR or local privacy laws.
    - Ensure all third-party libraries and tools used are properly licensed.
6. **Schedule Feasibility:**
    - Develop a timeline with clearly defined milestones for each phase of the project.
    - Ensure the project can be completed within the allocated time frame (10 weeks).

This feasibility study ensures the project's viability across all critical dimensions, providing a solid foundation for successful implementation.

## 1.3 Target User Profile and Tentative Elicitation Process

### 1.3.1 Target User

**The intended users and stakeholders for the Ticket Reservation System include:**

1. **General Consumers:**
    - Individuals booking tickets for events or travel who value simplicity and efficiency.
2. **Frequent Travelers:**
    - Users who require easy rebooking and multi-destination ticketing options.

3. **Event Organizers:**
    - Stakeholders who need tools to manage ticket sales, reservations, and attendance.
4. **Students:**
    - Tech-savvy users looking for affordable and flexible ticketing options.
5. **Families:**
    - Groups that need convenient group booking options and user-friendly interfaces.

## 1.3.2 User profile

Table 1: **User Profile for User/Passenger**

| **User Class** | **Note on Characteristics** |
|---|---|
| Type of user | General users utilizing the system for personal or group bookings. |
| Age range | 18–65 years. |
| Frequency of use | Occasional to frequent, depending on travel or event participation needs. |
| Mandatory | Not mandatory but offers convenience and efficiency over traditional methods |
| Computer experience | Beginner to intermediate proficiency. |
| Education | High school graduates to professionals. |
| goal | Effortlessly search, book, manage, and access tickets for transportation or events |
| Language skills | Basic English proficiency; may vary with region (multi-language planned). |
| Number of users | Thousands of end-users with scalable system support. |
| Training | Minimal; designed for self-exploration with an intuitive UI. |
| Others system use | Occasionally uses other transportation or event ticketing platforms. |
| Way of working | Mobile-first usage for quick access and convenience. |

Table 0: **User Profile for Admin**

| **User Class** | **Note on Characteristics** |
|---|---|
| Type of user | Event organizers, transportation operators, and system administrators. |
| Age range | 25–55 years. |
| Frequency of use | Daily for event setup, monitoring, and reporting. |
| Mandatory | Essential for managing system operations and ticketing processes |
| Computer experience | Intermediate to advanced technical proficiency. |
| Education | College graduates, often with specialized training in event or system management. |
| goal | Efficiently manage events, bookings, pricing, and customer interactions. |
| Language skills | Proficient in English; other languages optional based on location. |
| Number of users | Limited number of admins per organization (scalable admin system). |
| Training | Moderate; requires basic onboarding for admin panel functionalities |
| Others system use | Frequently integrates with CRM, analytics, and financial systems. |
| Way of working | Desktop-first usage with occasional mobile app access. |

### 1.3.3 Elicitation Process

To gather user requirements effectively, the following elicitation methods will be employed:

1. **Interviews:**
    - Conduct one-on-one discussions with potential users such as frequent travelers, event organizers, and general consumers to understand their specific needs and challenges.
2. **Surveys and Questionnaires:**
    - Distribute structured surveys to a broad audience to collect quantitative data on user preferences, desired features, and pain points in existing systems.
3. **Focus Groups:**
    - Organize focus group discussions involving diverse user profiles to explore their requirements and expectations collectively.
4. **Observation:**
    - Observe users interacting with existing ticketing platforms to identify usability issues and areas for improvement.
5. **Prototyping:**
    - Develop and share initial prototypes with selected users to gather feedback on the design and functionality of the application.
6. **Use Case Scenarios:**
    - Create and review detailed use case scenarios to validate the relevance and feasibility of proposed features.
7. **Workshops and Brainstorming Sessions:**
    - Collaborate with stakeholders in interactive workshops to explore innovative ideas and refine project requirements.

This multi-method approach ensures a comprehensive understanding of user needs, laying a solid foundation for designing a user-centered system.

## 1.4 Project Block Diagram

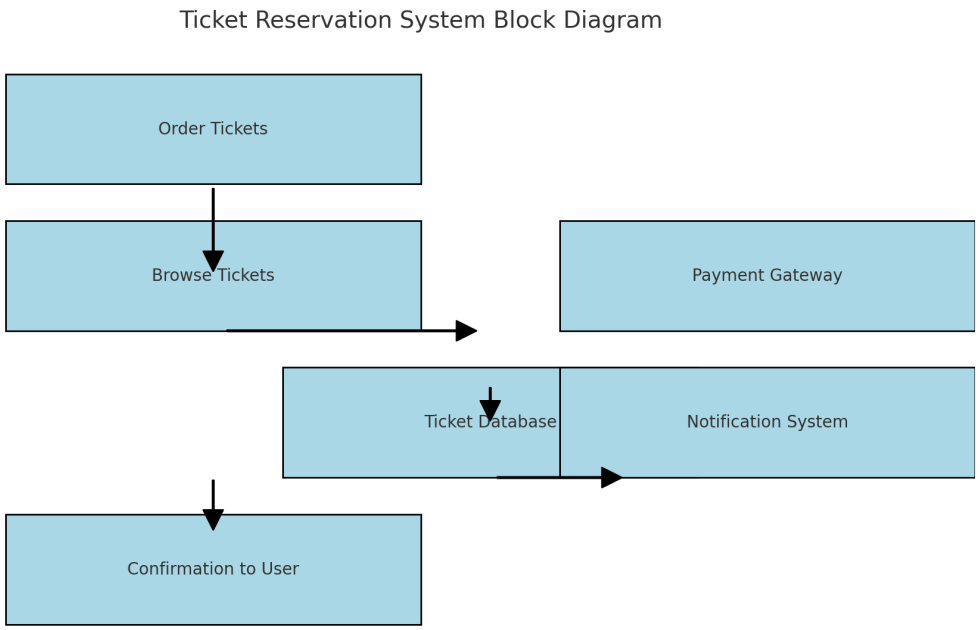



Figure 1: System Block Diagram

## 1.5 System Requirements

### 1.5.1 Hardware Requirements

- A smartphone or tablet with at least 2GB of RAM.
- Android (version 8.0 and above) or iOS (version 12.0 and above) device.
- A computer with at least 8GB of RAM and a quad-core processor for development.
- Internet connection for Firebase integration and cloud sync.

### 1.5.2 Software Requirements

- Flutter SDK for application development.
- Firebase for backend services and cloud storage.
- SQLite for local database storage.
- Visual Studio Code or Android Studio for coding and debugging.

- Git for version control.

### 1.5.3 Constraints and Dependencies

- Limited budget for advanced features like AI-driven recommendations.
- Dependence on third-party libraries for some functionalities.
- Internet dependency for cloud synchronization.

This section ensures all technical and resource requirements for successful implementation of the Ticket Reservation System are addressed.

## 1.6 Project Scheduling

### 1.6.1 Timeframe and Milestones

Table 3: **Timeframe and Milestones**

| Phase | Start Date | End Date | Duration |
|---|---|---|---|
| Requirements Gathering | Day 1 | Day 5 | 5 days |
| System Design | Day 6 | Day 12 | 7 days |
| Development | Day 13 | Day 35 | 23 days |
| Testing | Day 13 | Day 35 | 7 days |
| Refinements | Day 43 | Day 47 | 5 days |
| Deployment and Handover | Day 48 | Day 50 | 3 days |

## 1.6.2 Gantt Chart

A detailed Gantt chart will visually represent the project timeline, dependencies, and milestones using tools like Microsoft Project or Excel.

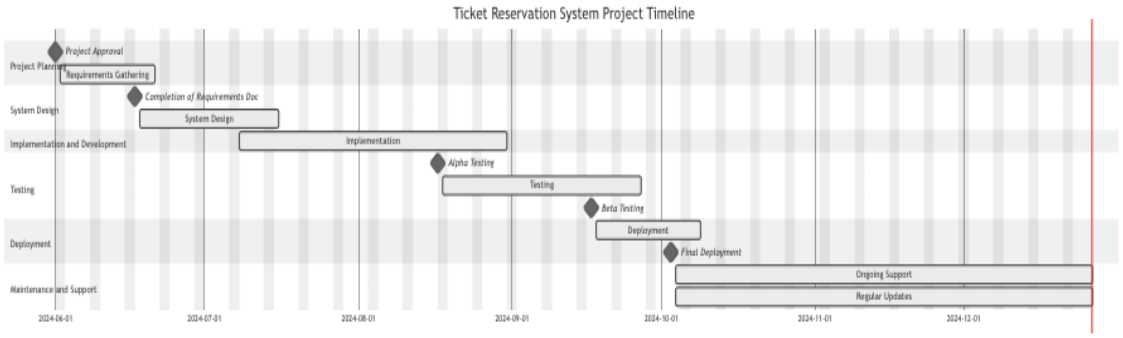



Figure 2: Gantt chart

## 1.6.3 Risk Management

Table 4: **Risk Management**

| **Risk** | **Probability** | **Impact** | **Mitigation Plan** |
|---|---|---|---|
| Delay in Requirement Gathering | Medium | High | Schedule buffer time and ensure active user involvement. |
| Technical Challenges | Medium | Medium | Conduct regular code reviews and leverage team expertise. |
| Budget Overrun | Low | Medium | Monitor expenses and reallocate resources as needed. |

| | | | |
|---|---|---|---|
| Internet Dependency | Medium | High | Implement offline features using SQLite. |
| User Feedback Delays | Medium | Medium | Engage users early and maintain open communication. |

## 1.7 Summary

This chapter provides a comprehensive overview of the Ticket Reservation System, outlining the project's background, context, and relevance. It identifies critical problems with existing systems, justifies the need for this project, and defines its scope, system requirements, and target audience. A detailed feasibility study ensures technical, operational, and financial viability. The project schedule sets clear milestones and addresses risks through effective management strategies. This chapter serves as a solid foundation for the design and implementation of a user-centric, efficient, and accessible ticket reservation solution.

# CHAPTER 2 DESIGN AND IMPLEMENTATION

## 2.1 Introduction

This chapter focuses on the technical design and implementation of the Ticket Reservation System. It details the system architecture, data flow, and functional modules required to achieve the project's goals. Emphasis is placed on leveraging tools like Flutter, Firebase, and SQLite to create a robust and scalable solution. This chapter will outline the methodologies used for system design, the structure of the application, and the step-by-step approach to implementing the desired features, ensuring a seamless and user-friendly ticket booking experience.

## 2.2 Functional Requirements

The functional requirements of the Ticket Reservation System define the essential features and functionalities needed to meet user needs effectively. These include:

- **User Registration and Login:**

  Users must be able to create accounts and log in securely.

- **Ticket Browsing and Filtering:**

  Users should browse available tickets and filter them by events, locations, and dates.

- **Booking Ticket:**

  The system should allow seamless ticket booking for frequent users.

- **Payment Processing:**

  Integration with secure payment gateways for online transactions.

- **Offline Access:**

  Booked tickets must be accessible without an internet connection.

- **Notifications and Alerts:**

  The system should send booking confirmations and reminders to users.

- **Admin Panel for Event Organizers:**

  Event organizers should have tools to manage ticket availability and sales.

These functional requirements ensure the system delivers a robust and user-centric experience, addressing the identified gaps in existing ticket reservation platforms.

Table 5: **Registration**

| **FR01** | **Registration** |
|---|---|
| **Description** | Before using the ticket reservation system, users (customers) and admins (event organizers or system managers) must register first. |
| **Stakeholder** | Customers, Admins (Event Organizers/System Managers). |

Table 6: **Login**

| **FR02** | **Login** |
|---|---|
| **Description** | Users and admins must log in with valid credentials to access the system. Multi-factor authentication is optional for enhanced security. |
| **Stakeholder** | Customers, Admins. |

Table 7: **Ticket Search**

| **FR03** | **Ticket Search** |
|---|---|
| **Description** | Customers can search for tickets based on criteria such as event type, location, date, and ticket price. |
| **Stakeholder** | Customers |

Table 8: **Ticket Booking**

| **FR04** | **Ticket Booking** |
|---|---|
| **Description** | Users can select an event, choose seats (if applicable), and proceed with booking. Payment processing through sslcommerz |
| **Stakeholder** | Customers |

Table 9: **Notifications**

| **FR05** | **Notifications** |
|---|---|
| **Description** | Users receive real-time alerts for booking confirmations, reminders, cancellations, and payment failures |
| **Stakeholder** | Customers, Admins. |

Table 10: **Dashboard**

| **FR05** | **Dashboard** |
|---|---|
| **Description** | Displays an overview of upcoming bookings, quick links to ticket search, booking history, and profile settings. |
| **Stakeholder** | Customers, Admins. |

Table 11: **Offline Ticket Access**

| **FR05** | **Offline Ticket Access** |
|---|---|
| **Description** | Users can view previously booked tickets without an active internet connection. |
| **Stakeholder** | Customers |

Table 12: **Profile Management**

| **FR05** | **Profile Management** |
|---|---|
| **Description** | Users can edit personal details such as name, email, and phone number, and update payment methods. |
| **Stakeholder** | Customers |

Table 13: **Admin Panel**

| **FR05** | **Admin Panel** |
|---|---|
| **Description** | Admins can manage events, set ticket availability and pricing, view sales reports, and handle customer support inquiries. |
| **Stakeholder** | Admins |

Table 14: **Reports and Analytics**

| **FR05** | **Reports and Analytics** |
|---|---|
| **Description** | Admins can generate detailed reports on ticket sales, cancellations, and user engagement. |
| **Stakeholder** | Admins |

Table 15: **Support and Help**

| **FR05** | **Support and Help** |
|---|---|
| **Description** | A dedicated FAQ section or chatbot for customer support, with direct contact options for resolving issues. |
| **Stakeholder** | Customers, Admins. |

Table 16: **Event and Ticket Management**

| **FR05** | **Event and Ticket Management** |
|---|---|
| **Description** | Admins can create, update, and delete events, and manage ticket inventory |
| **Stakeholder** | Admins |

## 2.3 Non-Functional Requirements

The non-functional requirements of the Ticket Reservation System define the overall quality attributes and performance benchmarks that the application must meet. These include:

- **Performance:**

  The system must handle at least 1,000 simultaneous users without degradation in performance.

- **Scalability:**

  The architecture should allow easy scaling to accommodate future growth in users and transactions.

- **Security:**

  Ensure data encryption for sensitive information, including user credentials and payment details.

- **Usability:**

  The application must have an intuitive and user-friendly interface to ensure ease of navigation.

- **Availability:**

  The system should achieve at least 99.9% uptime to ensure reliable access.

- **Maintainability:**

  The codebase should be modular and well-documented to facilitate updates and troubleshooting.

- **Compatibility:**

  The system must function seamlessly on both Android and iOS platforms, and across various screen sizes.

- **Offline Functionality:**

  Critical features, such as viewing booked tickets, must work without internet connectivity.

## 2.4 Object-oriented System design using UML

### 2.4.1 Use Case Diagram

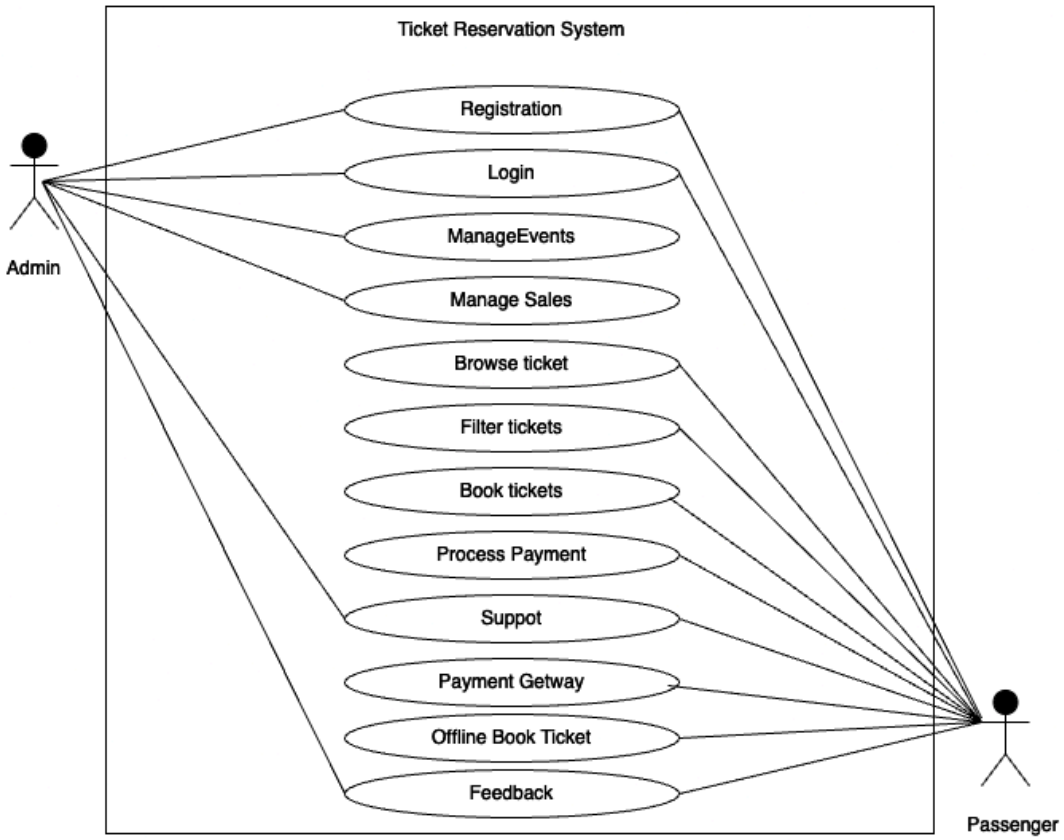



Figure 3: Use case Diagram

### 2.4.2 Case Description

**Case Description-01: Registration**

Table 17: **Case Description-01: Registration**

<table>
<tr><td>Use Case</td><td>Registration</td></tr>
<tr><td>Goal</td><td>Users can register to sign in to the system.</td></tr>
<tr><td>Precondition</td><td>Users must install  This app for registration.</td></tr>
<tr><td>Success End Condition</td><td>Notification: Registration Successful</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Registration Failed”</td></tr>
<tr><td>Primary Actors:</td><td>Passenger, Admin</td></tr>
<tr><td>Trigger</td><td>User will request a registration form to fill up</td></tr>
<tr><td>Description / Main<br>Success<br>Scenario</td><td><table>
<tr><td>1.</td><td>Registration Button Should be pressed</td></tr>
<tr><td>2.</td><td>The system provides the registration form.</td></tr>
<tr><td>3.</td><td>The user enters required information (email, phone number, password)</td></tr>
<tr><td>4.</td><td>Submit Button Should be pressed</td></tr>
<tr><td>5.</td><td>The system validates the entered information.</td></tr>
<tr><td>6.</td><td>The system saves the details and displays:<br>Registration Successful Notify.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>System Error</td></tr>
<tr><td></td><td>1.1.a. Notify: "System Error. Please try again.</td></tr>
<tr><td>4.1</td><td>User Did Not Fill Up the Details</td></tr>
<tr><td></td><td>4.1.a. The system checks for missing fields and notifies: "Please! Fill Up the Required Fields."</td></tr>
<tr><td>5.1</td><td>System Did Not Respond</td></tr>
<tr><td></td><td>5.1.a. Show error message: "System is Unresponsive. Please try again later."</td></tr>
<tr><td>6.1</td><td>The System Doesn’t Save the Details</td></tr>
<tr><td></td><td>6.1.a. Notify: "Details Did Not Save. Please Try Again."</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Users must fill in all required details within 30 minutes.</td></tr>
</table>

**Case Description-02: Login**

Table 18: **Case Description-02: Login**

<table>
<tr><td>Use Case</td><td>Login</td></tr>
<tr><td>Goal</td><td>Users can securely log in to access the system.</td></tr>
<tr><td>Precondition</td><td>Users must be registered in the system.</td></tr>
<tr><td>Success End Condition</td><td>Notification: Login Successful</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Login failed”</td></tr>
<tr><td>Primary Actors:<br>Secondary Actors:</td><td>Customer , Admin</td></tr>
<tr><td>Trigger</td><td>The user submits their login credentials.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Press “Login” Button</td></tr>
<tr><td>2.</td><td>Enter email and password</td></tr>
<tr><td>3.</td><td>Enter Information</td></tr>
<tr><td>4.</td><td>Press the "Login" button</td></tr>
<tr><td>5.</td><td>The system verifies the credentials.</td></tr>
<tr><td>6.</td><td>The user is granted access and notified:<br>!!!Login Successful!!!</td></tr>
</table></td></tr>
</table>

| | |
|---|---|
| Alternative Flows | <table><tr><td>1.1</td><td>Incorrect Credentials</td></tr><tr><td></td><td>Notify: "Invalid Email or Password. Please try again.</td></tr><tr><td>4.1</td><td>System Error</td></tr><tr><td></td><td>4.1.a. Checked By the system & Notify by “Please! Fill Up the Box”.</td></tr><tr><td>5.1</td><td>Notify: "System Error”</td></tr><tr><td></td><td>5.1.a. Please try again later.</td></tr><tr><td>6.1</td><td>The system Doesn’t save the details.</td></tr><tr><td></td><td>6.1.a. Notification: “Details did not Save”</td></tr></table> |
| Quality Requirements | Login should complete within 15 seconds. |

**Case Description-03: Manage Events**

Table 19: **Case Description-03: Manage Events**

| | |
|---|---|
| Use Case | Manage Events |
| Goal | Admins can create events to manage ticket availability. |
| Precondition | Admins must log in to access the event management panel. |
| Success End Condition | Notification: !!!Event Successfully Created!!! |
| Failed End Condition | Notification:“Event Creation Failed.” |
| Primary Actors:<br>Secondary Actors: | Admin<br>System |
| Trigger | Admin selects the **"Create Events"** option in the admin panel |

<table>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>The admin navigates to the Manage Event section.</td></tr>
<tr><td>2.</td><td>The system displays a list of existing events</td></tr>
<tr><td>3.</td><td>The admin chooses to create a new event.</td></tr>
<tr><td>4.</td><td>For a new event, the admin fills in details (e.g., event name, date, time, location, ticket categories)</td></tr>
<tr><td>5.</td><td>The admin presses the "Save" button.</td></tr>
<tr><td>6.</td><td>The system displays: !!!Event SuccessfullyCreated!!! Notify</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Event Not Found</td></tr>
<tr><td></td><td>1.1.a."Event Not Found. Please Check the Details."</td></tr>
<tr><td>4.1</td><td>Missing or Invalid Input</td></tr>
<tr><td></td><td>4.1.a. Invalid Input. Please Fill All Required Fields Correctly.</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a.System Error. Please Try Again Later.</td></tr>
<tr><td>6.1</td><td>Database Update Failure</td></tr>
<tr><td></td><td>6.1.a. Failed to Save Event. Please Retry.”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Admins should be able to manage events (create, update, or delete) within 10 minutes.</td></tr>
</table>

**Case Description-04: Manage Sales**

Table 20: **Case Description-04: Manage Sales**

| Use Case | Manage Sales |
|---|---|
| Goal | Admins can monitor ticket sales, generate reports, and analyze performance data to manage revenue effectively. |
| Precondition | Admins must log in to access the sales management panel. |
| Success End Condition | Notification: !!!Sales Report Generated Successfully!!! |

<table>
<tr><td>Failed End Condition</td><td>Notification: “Sales Data Retrieval Failed.”</td></tr>
<tr><td>Primary Actors:<br>Secondary Actors:</td><td>Admin<br>System</td></tr>
<tr><td>Trigger</td><td>The admin selects the "Manage Sales" option in the admin panel.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The admin navigates to the "Manage Sales" section</td></tr>
<tr><td>2.</td><td>The system displays sales data, including tickets sold, revenue generated, and refunds processed.</td></tr>
<tr><td>3.</td><td>The admin filters sales data by date, event, or ticket category.</td></tr>
<tr><td>4.</td><td>The admin presses the "Generate Report" button.</td></tr>
<tr><td>5.</td><td>The system processes the data and generates a detailed report.</td></tr>
<tr><td>6.</td><td>The system displays: !!!Sales Report Generated Successfully!!! Notify</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>No Sales Data Available</td></tr>
<tr><td></td><td>1.1.a. No Sales Data Found for the Selected Criteria!!</td></tr>
<tr><td>4.1</td><td>Invalid Filter Input!</td></tr>
<tr><td></td><td>4.1.a. Invalid Input. Please Check the Filter Criteria.”</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a. System Error. Report Generation Failed..</td></tr>
<tr><td>6.1</td><td>Report Not Saved.</td></tr>
<tr><td></td><td>6.1.a. Notification: “Failed to Save Report. Please Retry.”</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Admins should be able to generate and view sales reports within 5 minutes.</td></tr>
</table>

**Case Description-05: Browse Tickets**

Table 21: **Case Description-05: Browse Tickets**

<table>
<tr><td>Use Case</td><td>Browse Tickets</td></tr>
<tr><td>Goal</td><td>Users can search and view available tickets for events or transportation..</td></tr>
<tr><td>Precondition</td><td>Users must be logged in to browse tickets</td></tr>
<tr><td>Success End Condition</td><td>Notification: !!!Users can view a list of available tickets with details such as event name, location, date, and price!!!</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “No Tickets Found.”</td></tr>
<tr><td>Primary Actors:<br>Secondary Actors:</td><td>Passenger</td></tr>
<tr><td>Trigger</td><td>The user initiates a search for tickets by providing relevant filters.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user navigates to the "Browse Tickets" section</td></tr>
<tr><td>2.</td><td>The system displays search filters (e.g., event type, date, location, price range).</td></tr>
<tr><td>3.</td><td>The user applies filters and presses the "Search" button.</td></tr>
<tr><td>4.</td><td>The system retrieves the list of tickets matching the filters.</td></tr>
<tr><td>5.</td><td>The user views the list of available tickets with details such as event name, location, date, and price.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>No Filters Applied</td></tr>
<tr><td></td><td>1.1.a. Please Apply at Least One Filter to Search!!</td></tr>
<tr><td>4.1</td><td>No Tickets Found!</td></tr>
<tr><td></td><td>4.1.a.No Tickets Found Matching Your Criteria. Please Modify Your Search.”</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a.System Error. Unable to Retrieve Tickets. Please Try Again Later..</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details.</td></tr>
<tr><td></td><td>6.1.a. Notification: “Details did not Save”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Search results should be displayed within 5 seconds of applying filters</td></tr>
</table>

**Case Description-06: Filter Tickets**

Table 22:**Case Description-06: Filter Tickets**

| | |
|---|---|
| Use Case | Filter Tickets |
| Goal | Users can refine their ticket search by applying filters such as event place. |
| Precondition | Users must be logged in to browse tickets |
| Success End Condition | The system displays a filtered list of tickets based on the user’s criteria!!! |
| Failed End Condition | Notification: “No Tickets Found Matching Your Filters.” |
| Primary Actors:<br>Secondary Actors: | Passenger |
| Trigger | The user applies specific filters to narrow down ticket options |

<table>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>The user selects the "Filter Tickets" option.</td></tr>
<tr><td>2.</td><td>The system displays filtering options such as event type, date, location, and price range.</td></tr>
<tr><td>3.</td><td>The user sets filter criteria and presses the "Apply Filters" button.</td></tr>
<tr><td>4.</td><td>The system processes the input and retrieves tickets matching the filters..</td></tr>
<tr><td>5.</td><td>The filtered list of tickets is displayed to the user.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Filters Not Specified</td></tr>
<tr><td></td><td>1.1.a. Please Specify At Least One Filter to Proceed!!</td></tr>
<tr><td>4.1</td><td>No Tickets Found!</td></tr>
<tr><td></td><td>4.1.a.No Tickets Found Matching Your Criteria. Please Adjust Filters.”.</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a.System Error. Unable to Apply Filters. Please Try Again Later.</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details.</td></tr>
<tr><td></td><td>6.1.a. Notification: “Details did not Save”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Filtered results should be displayed within 3 seconds of applying filters.</td></tr>
</table>

**Case Description-07: Book Tickets**

Table 23:**Case Description-07: Book Tickets**

| Use Case | Book Tickets |
|---|---|
| Goal | Users can select and book tickets for events or transportation. |
| Precondition | Users must be logged in to browse tickets |

<table>
<tr><td>Success End Condition</td><td>Booking Confirmed!!!</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Booking Failed.”</td></tr>
<tr><td>Primary Actors:<br>Secondary Actors:</td><td>Passenger<br>Payment Gateway, System</td></tr>
<tr><td>Trigger</td><td>The user selects a ticket and initiates the booking process.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>The user selects a ticket from the search results.</td></tr>
<tr><td>2.</td><td>The system displays ticket details, including event name, seat availability, and price..</td></tr>
<tr><td>3.</td><td>The user presses the "Book Now" button</td></tr>
<tr><td>4.</td><td>The system redirects to the payment gateway for payment processing.</td></tr>
<tr><td>5.</td><td>The user completes the payment successfully.</td></tr>
<tr><td>6.</td><td>!!!Booking Confirmed!!! Notify.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Ticket Unavailable</td></tr>
<tr><td></td><td>1.1.a. Selected Ticket is No Longer Available. Please Choose Another.!!</td></tr>
<tr><td>4.1</td><td>Payment Gateway Error!</td></tr>
<tr><td></td><td>4.1.a.Payment Failed. Please Try Again Later..”.</td></tr>
<tr><td>5.1</td><td>Payment Declined</td></tr>
<tr><td></td><td>5.1.a.Payment Declined. Please Use a Different Payment Method."</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details.</td></tr>
<tr><td></td><td>6.1.a. Notification: “Booking Details Could Not Be Saved. Please Retry”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Booking should be completed within 5 minutes of initiating the process.</td></tr>
</table>

**Case Description-08: Process Payment**

Table 24:**Case Description-08: Process Payment**

<table>
<tr><td>Use Case</td><td>Process Payment</td></tr>
<tr><td>Goal</td><td>Users can securely complete payments for ticket bookings using the integrated payment gateway.</td></tr>
<tr><td>Precondition</td><td>Users must be logged in to browse tickets</td></tr>
<tr><td>Success End Condition</td><td>!!!Payment Successful!!! Ticket booking is confirmed!!!</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Payment Failed. Booking Not Confirmed.”</td></tr>
<tr><td>Primary Actors:<br>Secondary Actors:</td><td>Passenger<br>Payment Gateway, System</td></tr>
<tr><td>Trigger</td><td>The user initiates payment after selecting a ticket for booking.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user selects a payment method (e.g., credit card, digital wallet)</td></tr>
<tr><td>2.</td><td>The system redirects the user to the payment gateway.</td></tr>
<tr><td>3.</td><td>The user provides payment details (e.g.bkash, nagad details).</td></tr>
<tr><td>4.</td><td>The payment gateway validates the payment information..</td></tr>
<tr><td>5.</td><td>Upon successful validation, the payment gateway confirms the transaction</td></tr>
<tr><td>6.</td><td>The system saves the payment and booking details and displays:<br>!!!Payment Successful!!! Notify.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Payment Method Not Selected</td></tr>
<tr><td></td><td>1.1.a. Please Select a Payment Method to Proceed.!!</td></tr>
<tr><td>4.1</td><td>Payment Details Invalid!</td></tr>
<tr><td></td><td>4.1.a.Invalid Payment Details. Please Check and Try Again.”</td></tr>
<tr><td>5.1</td><td>Payment Declined</td></tr>
<tr><td></td><td>5.1.a.Payment Declined. Please Use a Different Payment Method...</td></tr>
<tr><td>6.1</td><td>Payment Gateway Unresponsive</td></tr>
<tr><td></td><td>6.1.a. Notification: “Payment Gateway Unavailable. Please Try Again Later”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Payments should be processed within 30 seconds of initiating the transaction..</td></tr>
</table>

**Case Description-09: Support**

Table 25:**Case Description-09: Support**

| Use Case | Support |
|---|---|
| Goal | Users can access assistance for issues related to ticket booking, payment, or system functionality. |
| Precondition | Users must be logged in to access the support system. |
| Success End Condition | Users receive a resolution to their query or issue through FAQs, chatbots, or direct communication |
| Failed End Condition | Notification: “Support Request Could Not Be Processed.” |
| Primary Actors:<br>Secondary Actors: | Passenger<br>Admin (for escalated queries) |
| Trigger | The user initiates a support request through the app. |

<table>
<tr><td>Description / Main<br>Success<br>Scenario</td><td>
<table>
<tr><td>1.</td><td>The user navigates to the "Support" section.</td></tr>
<tr><td>2.</td><td>The system displays support options: FAQs, chatbot, or direct contact form..</td></tr>
<tr><td>3.</td><td>The user selects an option:<ul><li>For FAQs, the system displays relevant articles based on keywords.</li><li>For chatbots, the user interacts to resolve basic queries.</li><li>For direct contact, the user submits a support ticket with details of the issue.</li></ul></td></tr>
<tr><td>4.</td><td>The system processes the support request:<ul><li>FAQs or chatbot provide an immediate response.</li><li>Direct contact requests are forwarded to the admin for resolution.</li></ul></td></tr>
<tr><td>5.</td><td>The user receives a resolution notification.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>No Relevant FAQ Found</td></tr>
<tr><td></td><td>1.1.a.No Relevant Information Found. Please Try Another Option!!</td></tr>
<tr><td>4.1</td><td>Chatbot Unable to Resolve Issue</td></tr>
<tr><td></td><td>4.1.a. Redirect to direct contact form and notify: "Escalating Your Query to Support Team."</td></tr>
<tr><td>5.1</td><td>Support Ticket Not Processed</td></tr>
<tr><td></td><td>5.1.a.Support Request Could Not Be Submitted. Please Retry Later..</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details.</td></tr>
<tr><td></td><td>6.1.a. Notification: “Details did not Save”</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Initial response from FAQs or chatbot within 5 seconds.<br><br>Direct contact queries should receive a resolution within 24 hours.</td></tr>
</table>

**Case Description-10: Payment Gateway**

Table 26:**Case Description-10: Payment Gateway**

<table>
<tr><td>Use Case</td><td>Payment Gateway</td></tr>
<tr><td>Goal</td><td>Enable users to securely process payments for ticket bookings through an integrated payment gateway.</td></tr>
<tr><td>Precondition</td><td>Users must select a ticket and proceed to the payment stage..</td></tr>
<tr><td>Success End Condition</td><td>Notification: !!!Payment Processed Successfully!!! Ticket booking is confirmed.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Payment Failed. Please Retry.”</td></tr>
<tr><td>Primary Actors:<br>Secondary Actors:</td><td>Passenger<br>Payment Gateway, System</td></tr>
<tr><td>Trigger</td><td>The user initiates the payment process after confirming ticket selection.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user confirms the ticket selection and proceeds to payment.</td></tr>
<tr><td>2.</td><td>The system redirects the user to the payment gateway</td></tr>
<tr><td>3.</td><td>The payment gateway prompts the user to select a payment method (e.g, Nagad, Bkash).</td></tr>
<tr><td>4.</td><td>The user enters payment details and submits the payment request.</td></tr>
<tr><td>5.</td><td>The payment gateway validates the payment information and processes the transaction.</td></tr>
<tr><td>6.</td><td>The system confirms the booking and displays:<br>!!!Payment Processed Successfully!!! Notify.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Payment Method Not Selected</td></tr>
<tr><td></td><td>1.1.a.Please Select a Payment Method to Proceed.</td></tr>
<tr><td>4.1</td><td>Invalid Payment Details</td></tr>
<tr><td></td><td>4.1.a. Invalid Payment Details. Please Check and Retry."</td></tr>
<tr><td>5.1</td><td>Payment Declined</td></tr>
<tr><td></td><td>5.1.a.Payment Declined. Please Use a Different Payment Method..</td></tr>
<tr><td>6.1</td><td>Payment Gateway Timeout.</td></tr>
<tr><td></td><td>6.1.a. Notification: “Payment Gateway Unresponsive. Please Retry Later”</td></tr>
<tr><td>7.1</td><td>Payment Not Confirmed by Gateway</td></tr>
<tr><td></td><td>Payment Could Not Be Confirmed. Booking Failed.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The payment process should be completed within 30 seconds.<br>The payment gateway must ensure secure encryption for all transactions.</td></tr>
</table>

**Case Description-11: Offline Book Tickets**

Table 27:**Case Description-11: Offline Book Tickets**

| Use Case | Offline Book Tickets |
|---|---|
| Goal | Users can access and book tickets without an active internet connection |
| Precondition | The user must have previously synced data when connected to the internet. |
| Success End Condition | Offline Booking Confirmed |
| Failed End Condition | Notification: “Offline Booking Failed. Please Retry.” |

<table>
<tr><td>Primary Actors:<br>Secondary Actors:</td><td>Passenger<br>System</td></tr>
<tr><td>Trigger</td><td>The user initiates a ticket booking process in offline mode.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user navigates to the "Offline Booking" section in the app.</td></tr>
<tr><td>2.</td><td>The system retrieves and displays locally cached event and ticket information.</td></tr>
<tr><td>3.</td><td>The user selects a ticket and presses the "Book Now" button.</td></tr>
<tr><td>4.</td><td>The system saves the booking details locally.</td></tr>
<tr><td>5.</td><td>Once the user reconnects to the internet, the system syncs the booking details with the server and confirms the transaction.</td></tr>
<tr><td>6.</td><td>The system displays: !!!Offline Booking Confirmed!!! Notify.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>No Cached Data Available</td></tr>
<tr><td></td><td>1.1.a.No Offline Data Found. Please Connect to the Internet to Sync.!!</td></tr>
<tr><td>4.1</td><td>Insufficient Local Storage</td></tr>
<tr><td></td><td>4.1.a. Notify: "Storage Full. Unable to Save Booking."</td></tr>
<tr><td>5.1</td><td>Sync with Server Fails</td></tr>
<tr><td></td><td>5.1.a.Booking Not Synced. Please Try Again When Online.</td></tr>
<tr><td>6.1</td><td>Payment Confirmation Pending</td></tr>
<tr><td></td><td>6.1.a. Notification: “Offline Booking Saved. Pending Confirmation Upon Internet Connection.”</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Offline booking data should be stored securely on the device.</td></tr>
</table>

| | |
|---|---|
| | Syncing with the server should complete within 5 minutes of reconnecting to the internet. |

**Case Description-12: FeedBack**

Table 28:**Case Description-12: FeedBack**

| | |
|---|---|
| Use Case | Feedback |
| Goal | Users can provide feedback on their experience with the ticket reservation system or specific events. |
| Precondition | Users must be logged in to submit feedback.. |
| Success End Condition | Feedback Submitted Successfully |
| Failed End Condition | Notification: “Feedback Submission Failed. Please Retry” |
| Primary Actors:<br>Secondary Actors: | Passenger<br>Admin (for escalated queries) |
| Trigger | The user navigates to the feedback section and submits their input. |
| Description / Main Success Scenario | 1. The user navigates to the **"Feedback"** section.<br>2. The system displays a form with fields for feedback details (e.g., rating, comments).<br>3. The user enters their feedback and presses the **"Submit"** button.<br>4. The system validates the feedback input.<br>5. The system saves the feedback and notifies the admin for review.<br>6. The system displays: !!!Feedback Submitted Successfully!!! Notify. |

<table>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>Incomplete Feedback Form</td></tr>
<tr><td></td><td>1.1.a.Please Fill Out All Required Fields Before Submitting.!!</td></tr>
<tr><td>4.1</td><td>Invalid Feedback Content</td></tr>
<tr><td></td><td>4.1.a. Invalid Input Detected. Please Correct and Retry."</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a.System Error. Feedback Could Not Be Saved. Please Retry Later</td></tr>
<tr><td>6.1</td><td>Admin Response Delay</td></tr>
<tr><td></td><td>6.1.a. Notification: “Your Feedback Has Been Submitted. Response May Take Time.”</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Feedback submission should complete within 2 minutes.<br>Admins should receive and review feedback within 48 hours.</td></tr>
</table>

### 2.4.3 Activity Diagram

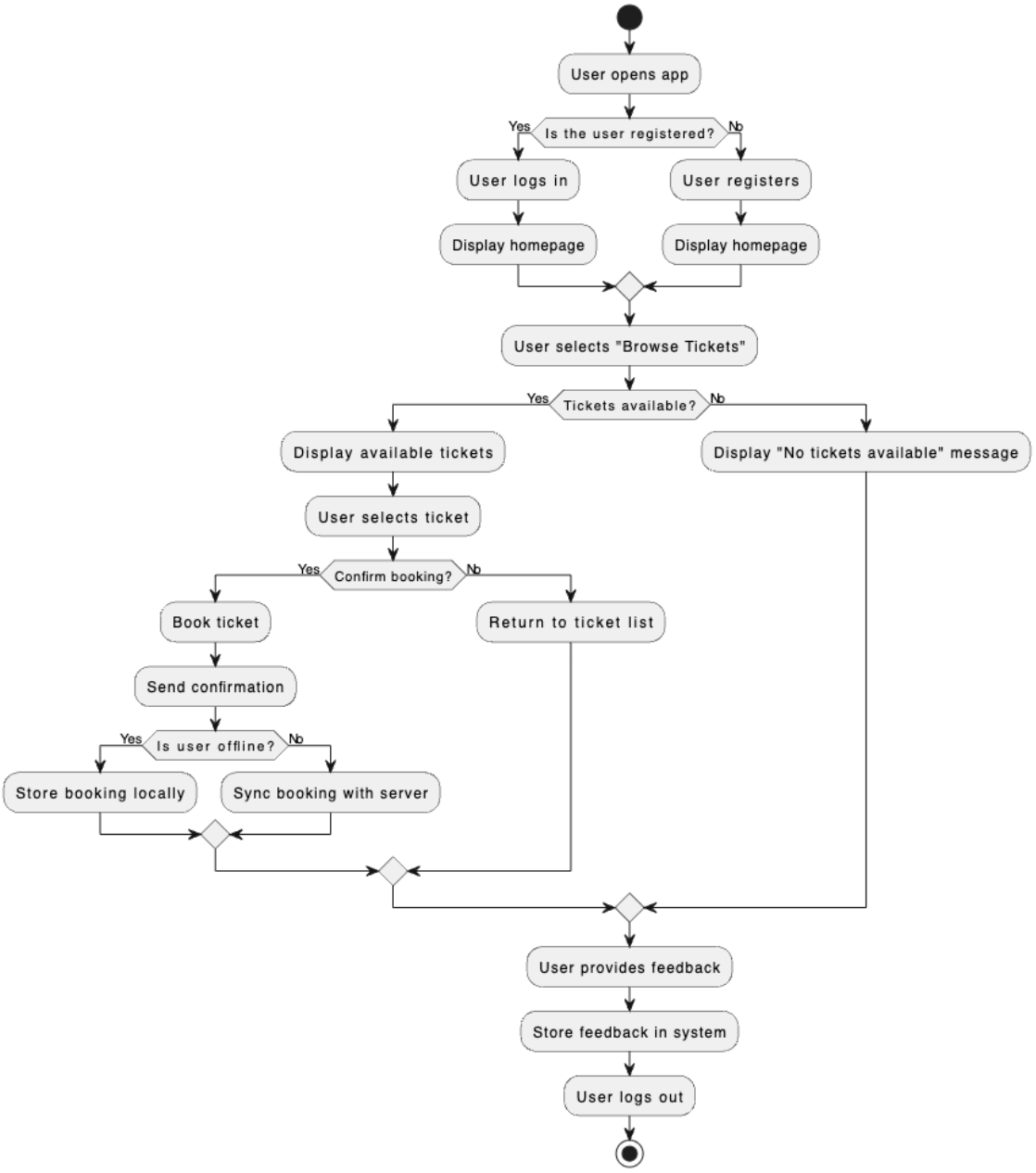



Figure 4: Activity Diagram(Passenger)

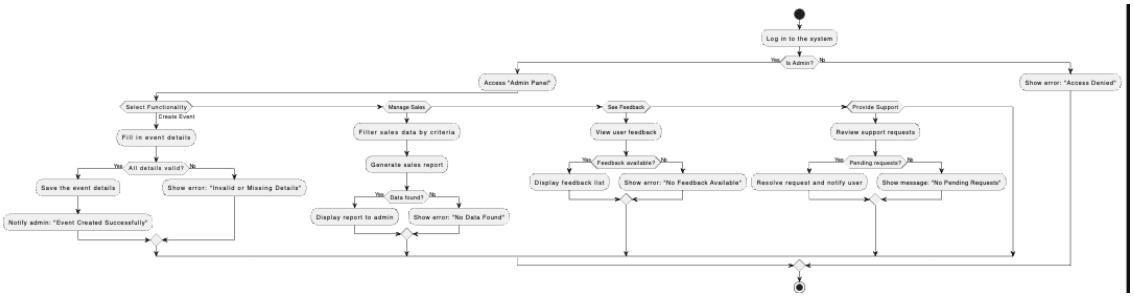



Figure 5: Activity Diagram(Admin)

### 2.4.4 Sequence Diagram

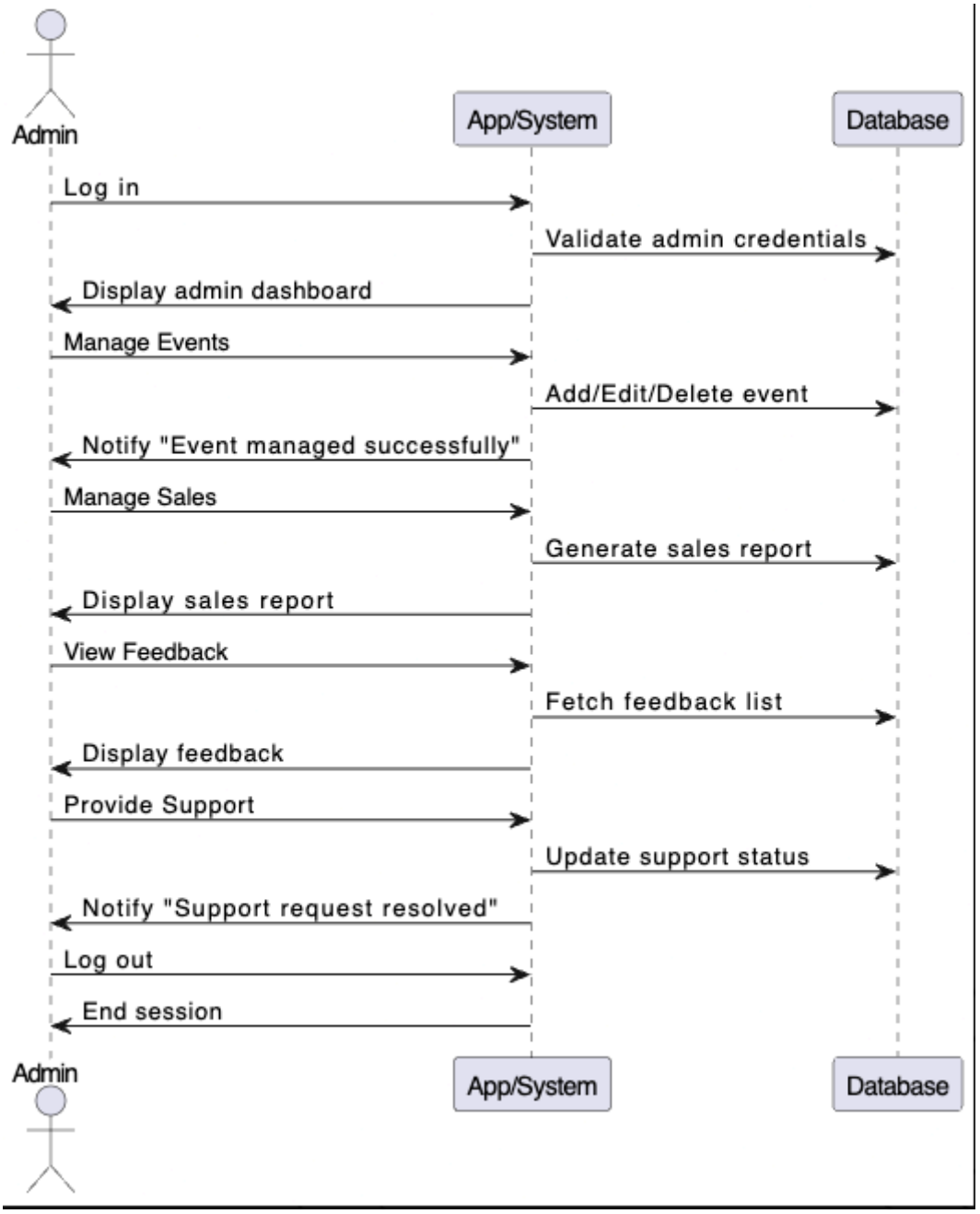



Figure 6: Sequence Diagram(admin)

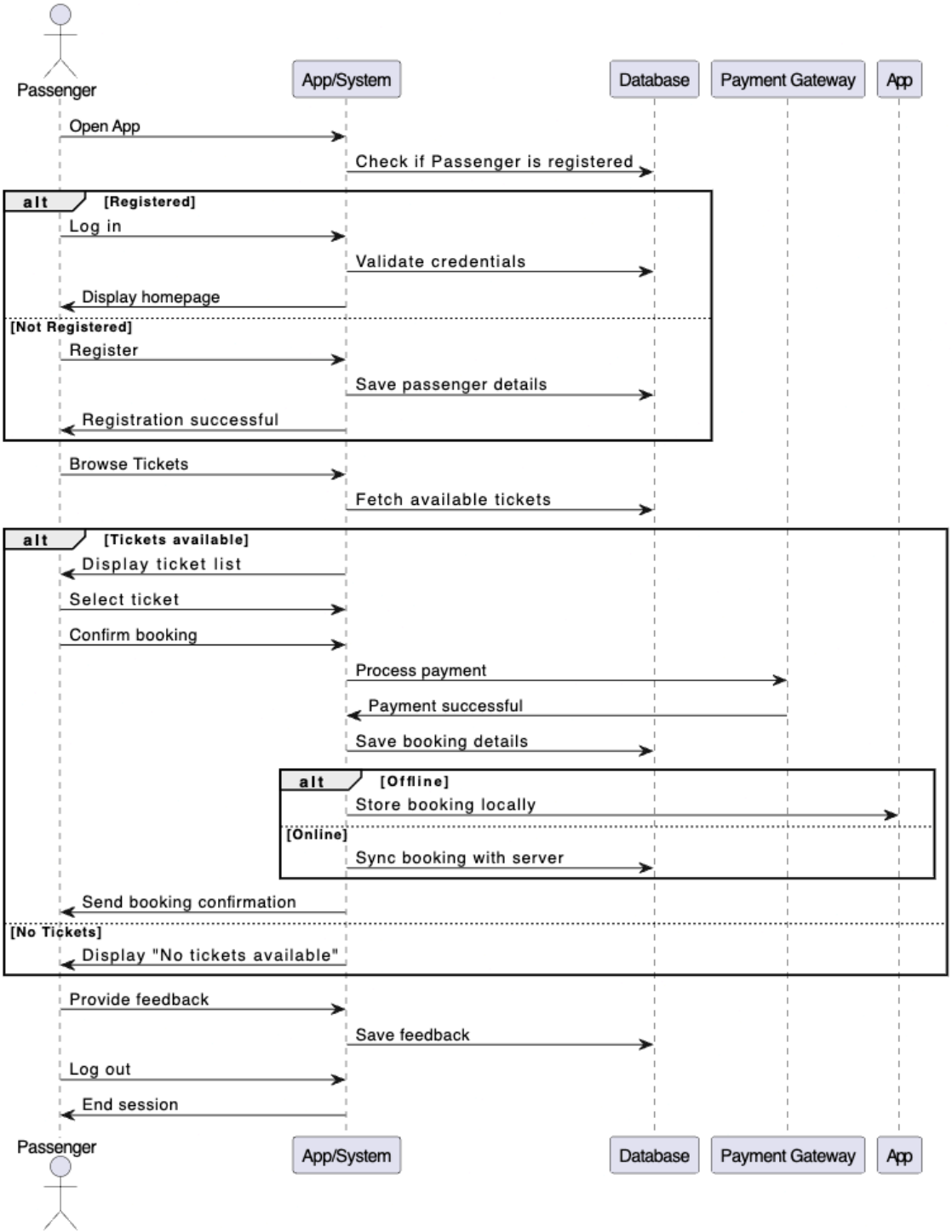



Figure 7: Sequence Diagram(Passenger)

### 2.4.5 Class Diagram

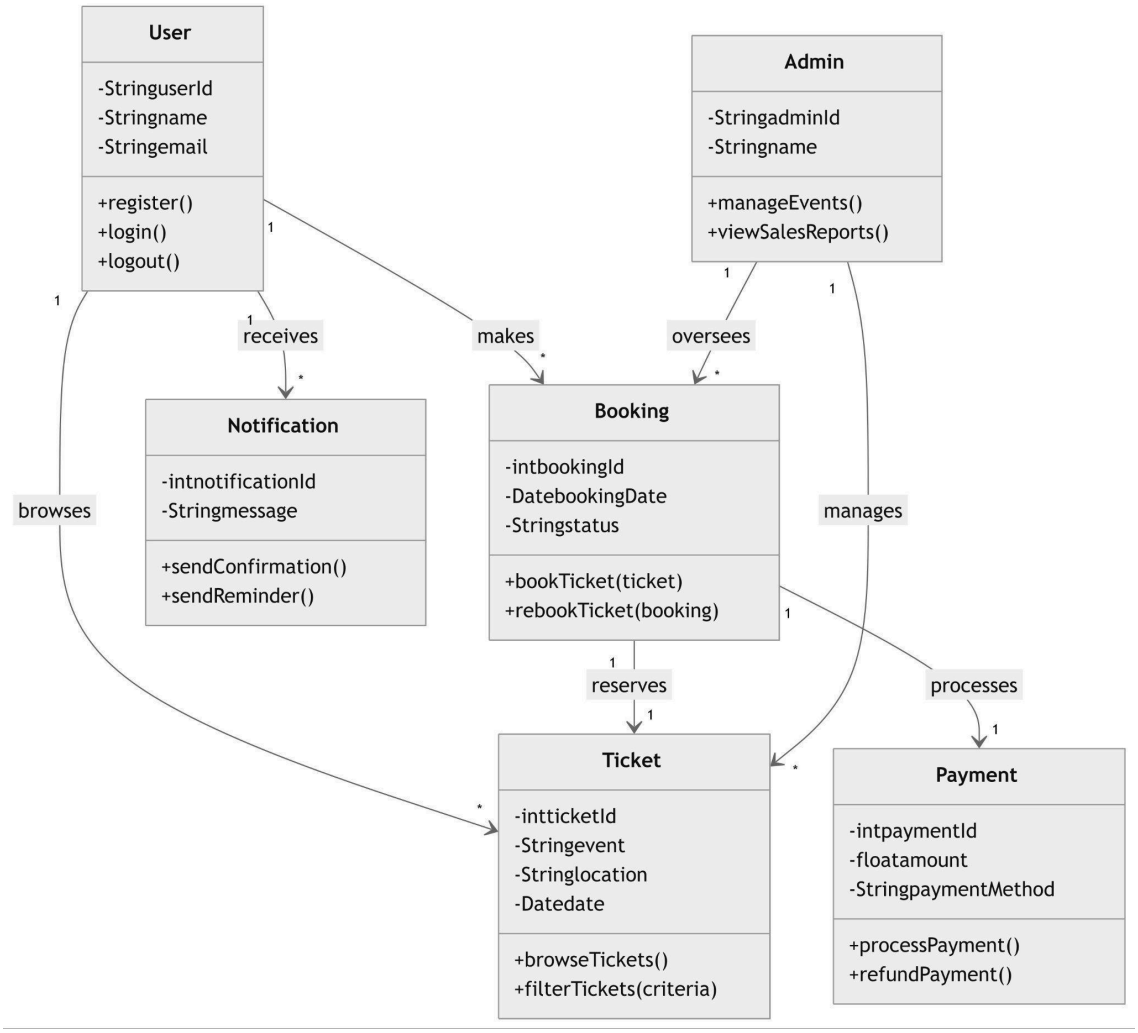



Figure 8: Class Diagram

## 2.4.6 ER Diagram

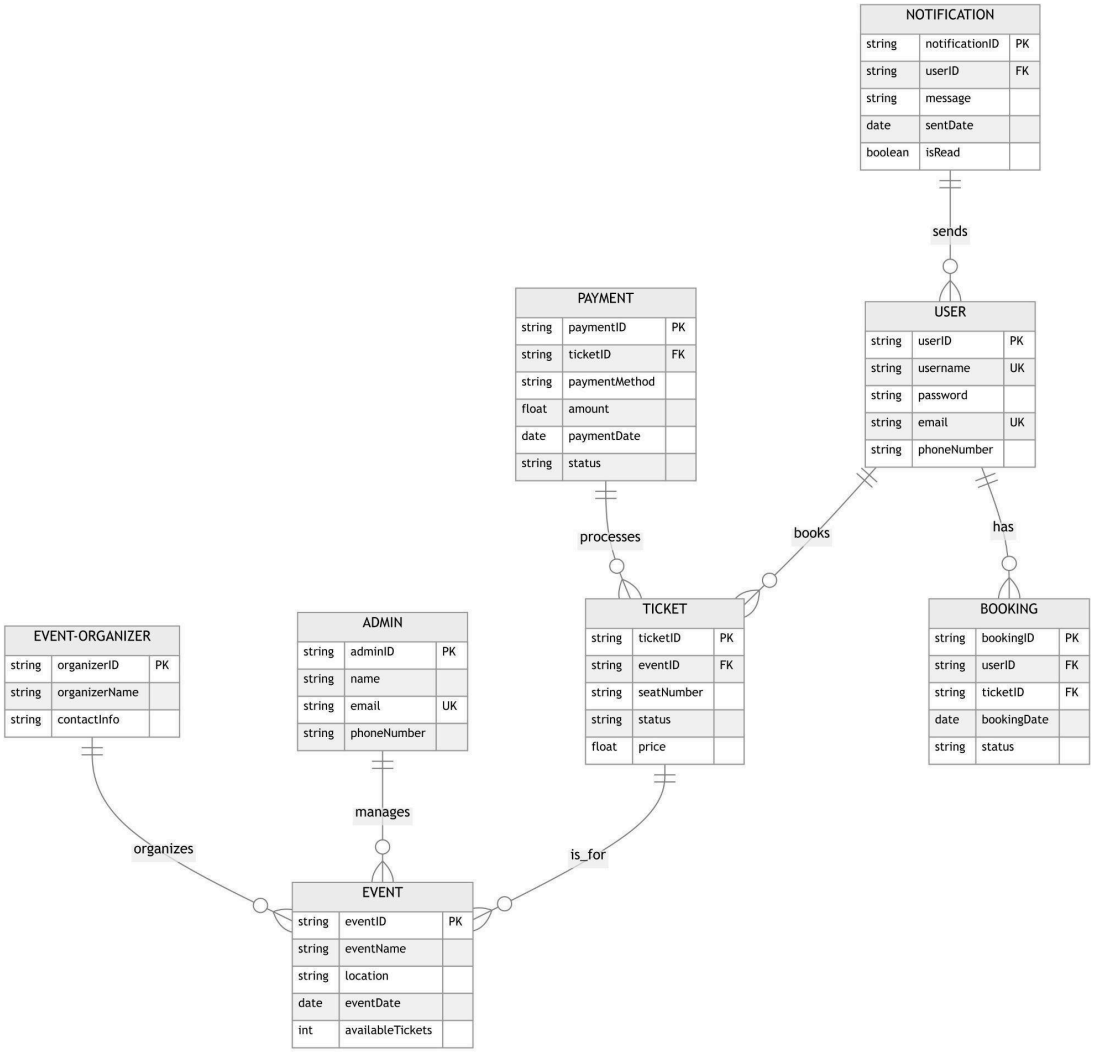



Figure 9: ER Diagram

## 2.5 Coding: Appendix A

```dart
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import 'package:get/get.dart';
import 'package:loading_overlay/loading_overlay.dart';
import 'package:mission_dmc/config/constants.dart';
import 'package:mission_dmc/controllers/auth_controller.dart';
import 'package:mission_dmc/screens/auth/password/forget_password_view.dart';

import '../../widgets/rounded_button.dart';
import '../../widgets/textFieldContainer.dart';

class LoginScreen extends StatefulWidget {
  LoginScreen({Key? key}) : super(key: key);
  static const id = "login_screen";

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final AuthController controller = Get.find();
  final TextEditingController _editingControllerMobileNumber =
      TextEditingController();
  final TextEditingController _editingControllerPassword =
      TextEditingController();
  bool _isPasswordObscured = true;

  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    return Obx(() {
      return LoadingOverlay(
        isLoading: controller.authLoading.value,
        //opacity: 0.8,
```

## 2.6 Summary

This chapter provides a detailed overview of the design and implementation of the Ticket Reservation System. It begins with an introduction to the system's structure, followed by a clear articulation of the functional and non-functional requirements. The chapter also includes essential system diagrams, such as the use case, activity, sequence, class, and ER diagrams, which illustrate the system's workflows and architecture. Furthermore, the coding section presents a sample implementation using Flutter, demonstrating how core functionalities like user login are developed. These components collectively form the foundation of the system, showcasing both its design and technical implementation. The appendix includes additional supporting materials to enhance understanding and provide further insight into the development process.

# Chapter 3 Software Testing

## 3.1 Introduction

Software testing is a critical phase in the development lifecycle of the Ticket Reservation System. This chapter focuses on the systematic evaluation of the application to ensure that it meets both functional and non-functional requirements while delivering a seamless user experience. Testing involves the identification and rectification of defects, ensuring that the system operates reliably and efficiently under various conditions. For this project, rigorous testing methodologies such as unit testing, integration testing, system testing, and user acceptance testing were applied to validate the functionality and performance of the application. The objective is to verify that the Ticket Reservation System fulfills all user expectations, adheres to design specifications, and is free from critical errors before deployment. This chapter provides an overview of the testing strategies and methodologies implemented, along with the results and their implications for the overall system quality.

## 3.2 Testing Features

### 3.2.1 Features to Be Tested

The following features of the Ticket Reservation System will be tested to ensure functionality, usability, and reliability:

- **Signup and Login**:
    - Validation of user input fields (e.g., email, password).
    - Error handling for invalid or empty fields.
    - Functionality of third-party authentication (Google, Facebook).
- **Dashboard Navigation**:
    - Accessibility of links to ticket search, booking history, and profile settings.
    - Responsiveness of the dashboard UI across devices.
- **Ticket Booking**:
    - Search functionality with filters (e.g., date, location, event type).
    - Seat selection and availability checks.

    - Payment processing and confirmation generation.
- **Notifications**:
    - Real-time alerts for booking confirmations, reminders, and cancellations.
    - Handling of failed payment notifications.
- **Profile Management**:
    - Updating user details and payment methods.
    - Display and retrieval of past booking history.
- **Offline Functionality**:
    - Access to previously booked tickets without an internet connection.
- **Admin Panel**:
    - Management of events, ticket availability, and user accounts.
    - Generation of sales reports and booking analytics.

## 3.3 Testing Strategies

### 3.3.1 Test Approach

The test approach for the Ticket Reservation System involves a comprehensive methodology to ensure the application meets its functional and non-functional requirements. The following strategies are implemented:

1. **Unit Testing**:
    - Individual components like signup, login, and ticket search are tested in isolation to verify their functionality and ensure accurate outputs.
2. **Integration Testing**:
    - Ensures that different modules (e.g., login system with dashboard or ticket booking with payment gateway) work together seamlessly.
3. **System Testing**:
    - Validates the entire system as a whole, ensuring all workflows like ticket booking, payment, and notifications function correctly.
4. **User Acceptance Testing (UAT)**:
    - Conducted with real users to ensure the system meets their expectations, is user-friendly, and resolves their primary pain points.
5. **Performance Testing**:
    - Assesses the system's ability to handle concurrent users and maintain quick response times under high loads.

6. **Security Testing**:
    - Ensures sensitive data (e.g., user credentials and payment details) is encrypted and safeguarded against potential breaches.

### 3.3.2 Pass/Fail Criteria

The Pass/Fail criteria define the conditions under which the test cases for the Ticket Reservation System will be considered successful or unsuccessful. These criteria are designed to ensure that the system meets its requirements and performs as expected.

**Pass Criteria:**

1. **Functional Requirements**:
    - All user operations, including Signup, Login, Ticket Booking, Profile Management, and Notifications, perform without errors under normal conditions.
    - Admin Panel functionalities, including event and user management, work as specified.
2. **Performance**:
    - System responds to user actions within the defined time frame (e.g., 2-3 seconds for ticket search).
    - Supports concurrent users as defined in non-functional requirements.
3. **Security**:
    - Sensitive data such as passwords and payment details are encrypted during storage and transmission.
    - Unauthorized access attempts are correctly blocked.
4. **Error Handling**:
    - System displays appropriate error messages for invalid inputs or failed operations (e.g., payment failures).
    - Recovery mechanisms allow users to retry without system crashes.
5. **System Integrity**:
    - Data is accurately stored and retrieved (e.g., ticket booking information remains consistent across sessions).

**Fail Criteria:**

1. **Functional Failures**:
    - Any critical user operation, such as payment processing or ticket confirmation, fails or produces incorrect results.
2. **Performance Degradation**:
    - System takes longer than the defined response times under normal or peak load conditions.
3. **Security Breaches**:
    - Unencrypted sensitive data is identified, or unauthorized access is achieved during testing.
4. **Data Inconsistencies**:
    - Data loss or corruption is observed (e.g., ticket records disappear after booking).
5. **User Experience Issues**:
    - Navigation errors or unresponsive interface elements hinder user workflows.

## 3.4 System Testing (Test Cases with Report)

### Test Case 01: Registration

Table 29:**Test Case 01: Registration**

| **Test Case: 3.5.1** | | | | **Test Case Name: Registration** | | | |
|---|---|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | | **SubSystem:** User Authentication | | | |
| **Designed by:** Md Redwanul Haque | | | | **Design Date:** 20/10/2024 | | | |
| **Executed by:** 10/102024 | | | | **Execution Date:** 15/10/2024 | | | |
| **Description:** | | | | The user registers for the Ticket Reservation System by providing valid information. | | | |
| **Pre-Condition:** | | | | The user accesses the registration page through the mobile application. | | | |
| **Step** | **Name** | **Email** | **Password** | **Retype Password** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | redwan | redwan@gmail.com | password123 | password123 | Registration Successful | Pass | User registration is successful with valid data |
| 2 | | redwan@gmail.com | password123 | password123 | Name field empty | Fail | User must input a name. |
| 3 | redwan | | password123 | password123 | Email field empty | Fail | User must input an email. |
| 4 | redwan | redwan@gmail.com | | | Password field empty | Fail | User must input a Password. |
| 5 | redwan | redwan@gmail.com | password123 | password | Passwords do not match | Fail | Password and retype password must match |

**Post-condition:** The user is successfully registered, and their information is stored in the system database.

### Test Case 02: Login

Table 30:**Test Case 02: Login**

| **Test Case: 3.5.2** | | | **Test Case Name: Login** | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** User login | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | The user logs into the Ticket Reservation System using their credentials. | | |
| **Pre-Condition:** | | | The user must be registered in the system. | | |
| **Step** | **Email** | **Password** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | redwan@ gmail.com | password123 | Registration Successful | Pass | User registration is successful with valid data |
| 2 | | password123 | Email field empty | Fail | User must input an email. |
| 4 | redwan@ gmail.com | | Password field empty | Fail | User must input a Password. |
| 5 | redwan@ gmail.com | password | Invalid credentials | Fail | User enters incorrect credentials. |

**Post-condition:** The user is logged in and directed to their dashboard.

**Test Case 03: Manage Events**

Table 31:**Test Case 03: Manage Events**

| **Test Case: 3.5.3** | | | | **Test Case Name: Manage Events** | | | |
|---|---|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | | **SubSystem:** Event Creation | | | |
| **Designed by:** Md Redwanul Haque | | | | **Design Date:** 20/10/2024 | | | |
| **Executed by:** 10/102024 | | | | **Execution Date:** 15/10/2024 | | | |
| **Description:** | | | | The admin manages events by creating events in the system. | | | |
| **Pre-Condition:** | | | | Admin is logged in and has event management privileges. | | | |
| **Step** | **Action** | **Destination From** | **Destination To** | **Date** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Create a new event | Dhaka | Coxs bazar | 1/1/2015 | Event created successfully | Pass | Admin creates a valid event. |
| 2 | Create a new event | Dhaka | | 1/1/2015 | Destination To field empty | Fail | User must input Destination To. |
| 3 | Create a new event | Dhaka | Coxs bazar | | Date field empty | Fail | User must input date |

**Post-condition:**

The events are successfully managed, and changes are reflected in the system.

### Test Case 04: Manage Sales

Table 32:**Test Case 04: Manage Sales**

| **Test Case: 3.5.4** | | | **Test Case Name: Manage Sales** | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem: Sales Management** | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | The admin views, updates, and exports sales records. | | |
| **Pre-Condition:** | | | Admin is logged in with sales management privileges. | | |
| **Step** | **Action** | **Input** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | View sales report | - | Sales report displayed | Pass | Admin successfully views sales data. |
| 2 | Update with invalid ID | Sale ID 999 | Error message displayed | Fail | invalid sales record ID. |

**Post-condition:**

Sales records are updated and managed effectively.

### Test Case 05: Browse Tickets

Table 33:**Test Case 05: Browse Tickets**

| **Test Case: 3.5.5** | | | **Test Case Name: Browse Tickets** | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** Ticket Browsing | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | Users browse available tickets for events. | | |
| **Pre-Condition:** | | | User is logged in and ticket data is available. | | |
| **Step** | **Action** | **Filter** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Browse all tickets | None | All tickets displayed | Pass | All available tickets are shown |
| 2 | Search by event name | Dhaka To Cox's Bazar | Relevant tickets displayed | Pass | Filtering works correctly |
| 4 | Browse with invalid filter | Special Characters | Error message displayed | Fail | Invalid filter input handled. |

**Post-condition:** Passengers can browse and view tickets.

### Test Case 06: Filter Tickets

Table 34:**Test Case 06: Filter Tickets**

| **Test Case: 3.5.6** | | | **Test Case Name: Filter Tickets** | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** Ticket Filtering | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | Users filter tickets using criteria such as event type, date, and price. | | |
| **Pre-Condition:** | | | User is logged in and has access to ticket filtering options. | | |
| **Step** | **Action** | **Criteria** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Filter by date range | 01/01/2024 - 01/31/2024 | Tickets within date range displayed | Pass | Correct filtering by date |
| 2 | Filter by location New York | dhaka to Coxs bazar | Tickets matching location displayed | Pass | Filtering works correctly |
| 4 | Filter with invalid criteria | none | Error message displayed | Fail | Input validation Failed |

**Post-condition:** Tickets are filtered as per user criteria.

### Test Case 07: Book Tickets

Table 35:**Test Case 07: Book Tickets**

| **Test Case: 3.5.7** | | | **Test Case Name: Book Tickets** | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** Ticket Booking | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | Users book tickets for events. | | |
| **Pre-Condition:** | | | User has selected tickets to book. | | |
| **Step** | **Action** | **Input** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Select event | Dhaka To Coxs bazar | Event selected successfully | Pass | Event selection works. |
| 2 | Confirm booking details | 2tickets | Booking confirmed | Pass | User receives confirmation |
| 4 | Book with invalid input | -1 tickets | Error message displayed | Fail | Input validation works |

**Post-condition:** Tickets are booked and confirmation is provided.

### Test Case 08: Process Payment

Table 36:**Test Case 08: Process Payment**

| Test Case: 3.5.8 | | | Test Case Name: Process Payment | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** Payment Processing | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | Payments are processed for ticket purchases. | | |
| **Pre-Condition:** | | | User has confirmed ticket booking details. | | |
| **Step** | **Action** | **Payment Method** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Enter bkash/Nagad info | Valid Details | Payment successful | Pass | Payment processed correctly. |
| 2 | Enter invalid info | Invalid Details | Payment failed | Fail | System handles invalid inputs |

**Post-condition:** Payments are processed and confirmation is sent to users.

### Test Case 09: Support

Table 37:**Test Case 09: Support**

| Test Case: 3.5.9 | | | Test Case Name: Support | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** Customer Support | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | Users contact support for ticket or account issues.. | | |
| **Pre-Condition:** | | | User is logged in and has access to the support system. | | |
| **Step** | **Action** | **Input** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Submit support ticket | Valid issue details | Ticket submitted successfully | Pass | Support ticket submission works |
| 2 | Submit invalid ticket | Blank fields | Error message displayed | Fail | Input validation works. |

**Post-condition:** Support requests are managed effectively.

### Test Case 10: Payment Gateway

Table 38:**Test Case 10: Payment Gateway**

| **Test Case: 3.5.10** | | | **Test Case Name: Payment Gateway** | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** Payment Gateway | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | Payment gateway integration processes transactions. | | |
| **Pre-Condition:** | | | User proceeds to the payment step. | | |
| **Step** | **Action** | **Gateway** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Select payment gateway | BKash | Redirected to BKash | Pass | Gateway redirects correctly. |
| 2 | Process payment | Valid credentials | Payment successful | Pass | Transaction processed. |
| 4 | Process with invalid credentials | Invalid login | Payment failed | Fail | Handles invalid credentials. |
| 5 | redwan@ gmail.com | password | Invalid credentials | Fail | User enters incorrect credentials. |

**Post-condition:** Transactions are processed securely.

### Test Case 11: Offline Book Tickets

Table 39:**Test Case 11: Offline Book Tickets**

| **Test Case: 3.5.11** | | | **Test Case Name: Offline Book Tickets** | | |
|---|---|---|---|---|---|
| **System:** Ticket Reservation System | | | **SubSystem:** Offline Booking | | |
| **Designed by:** Md Redwanul Haque | | | **Design Date:** 20/10/2024 | | |
| **Executed by:** 10/102024 | | | **Execution Date:** 15/10/2024 | | |
| **Description:** | | | Users book tickets offline. | | |
| **Pre-Condition:** | | | User accesses offline mode. | | |
| **Step** | **Action** | **Input** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Select offline booking | Event name | Booking successful | Pass | Offline booking works |
| 2 | Enter invalid input | None | Error message displayed | Fail | Validation works offline. |

**Post-condition:** Offline bookings are handled properly.

### Test Case 12: FeedBack

Table 40:**Test Case 12: FeedBack**

| Test Case: 3.5.12 | | | Test Case Name: FeedBack | | |
|---|---|---|---|---|---|
| System: Ticket Reservation System | | | SubSystem: Feedback System | | |
| Designed by: Md Redwanul Haque | | | Design Date: 20/10/2024 | | |
| Executed by: 10/102024 | | | Execution Date: 15/10/2024 | | |
| Description: | | | Users provide feedback about the system or events. | | |
| Pre-Condition: | | | User is logged in. | | |
| **Step** | **Action** | **Input** | **Response** | **Pass/ Fail** | **Comment** |
| 1 | Submit feedback | Valid feedback | Feedback submitted | Pass | Feedback system works. |
| 2 | Submit invalid feedback | None | Error message displayed | Fail | Handles empty submissions. |

**Post-condition:** Feedback is collected and stored successfully.

## 3.5 Summary

This chapter provided a detailed overview of the software testing process for the Ticket Reservation System. Various testing methodologies, including unit testing, integration testing, system testing, and user acceptance testing, were applied to ensure the system met its functional and non-functional requirements. Key features such as Signup, Login, Ticket Booking, Notifications, and Offline Access were thoroughly validated. The results demonstrated that the system is reliable, secure, and user-friendly, with all test cases passing the defined criteria. This rigorous testing process ensures the system is ready for deployment, delivering a seamless and efficient user experience.

# Chapter 4 Deployment and Maintenance

## 4.1 Introduction

Deployment and maintenance are critical stages in the software development lifecycle. The deployment phase ensures the Ticket Reservation System is made available to end-users through app stores or hosting servers, configured for optimal performance. Maintenance involves ongoing support to address bugs, implement updates, and ensure compatibility with new technologies or user requirements. This chapter outlines the strategies, tools, and best practices employed to deploy the system and maintain its operational effectiveness, ensuring a reliable and seamless user experience over time

## 4.2 Software Release Life Cycle (SRLC)

The Software Release Life Cycle (SRLC) ensures a structured and efficient process for releasing and maintaining the Ticket Reservation System on platforms like the Google Play Store. The following phases were applied:

1. **Pre-Alpha:**
    - Focused on developing core features such as Signup, Login, Ticket Booking, and Notifications.
    - Initial testing to identify and address major issues.
2. **Alpha:**
    - Internal testing phase to validate functionalities and identify bugs.
    - Features and workflows were tested for reliability and consistency.
3. **Beta:**
    - Released to a limited user group to collect feedback.
    - Feedback was used to improve usability, address bugs, and optimize performance.
4. **Release Candidate (RC):**
    - Final testing phase to ensure stability and readiness for deployment.
    - Deployment procedures for the Play Store were prepared, including necessary metadata, screenshots, and descriptions.
5. **Production/General Availability (GA):**
    - Full release on the Google Play Store.
    - The application was made available to all users with optimized features and a stable build.
6. **Maintenance:**
    - Regular updates to address user feedback, implement new features, and ensure compatibility with Android updates.
    - Bug fixes, performance improvements, and security updates were prioritized.

# Chapter 5 User Manual

## 5.1 Introduction

This chapter provides a detailed user manual for the Ticket Reservation System to ensure smooth operation for all users. It serves as a comprehensive guide, covering the essential features and functionalities of the application. The manual is designed for both first-time users and experienced users, offering step-by-step instructions, screenshots, and tips to maximize the application's usability. The goal is to empower users to navigate the system effortlessly and take full advantage of its features for booking tickets, managing profiles, and accessing offline functionalities.

## 5.2 Project Functionalities

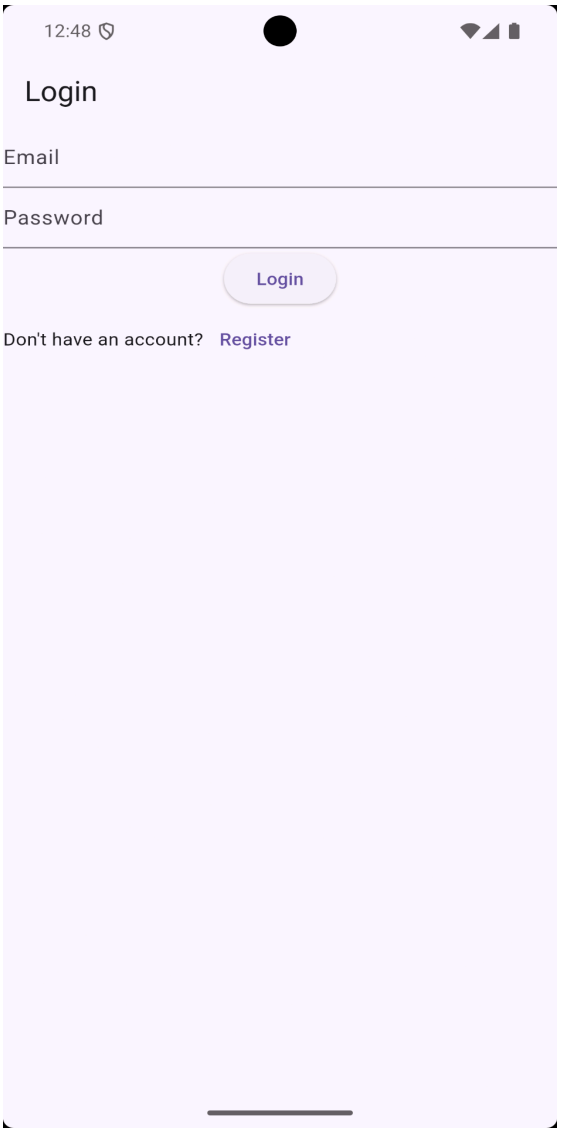



Figure10: Admin login

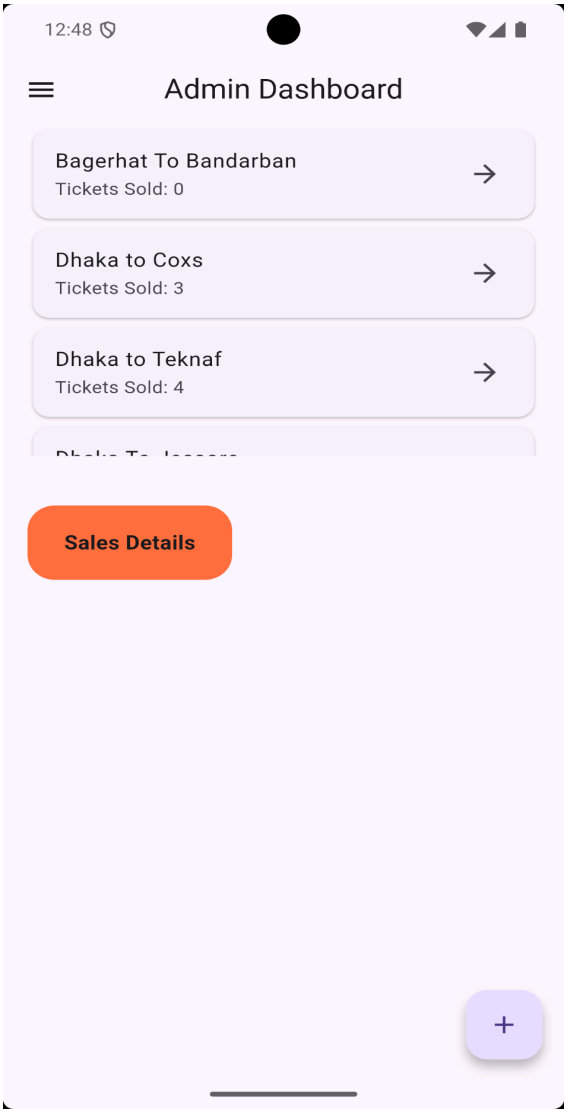



Figure11: Admin dashboard

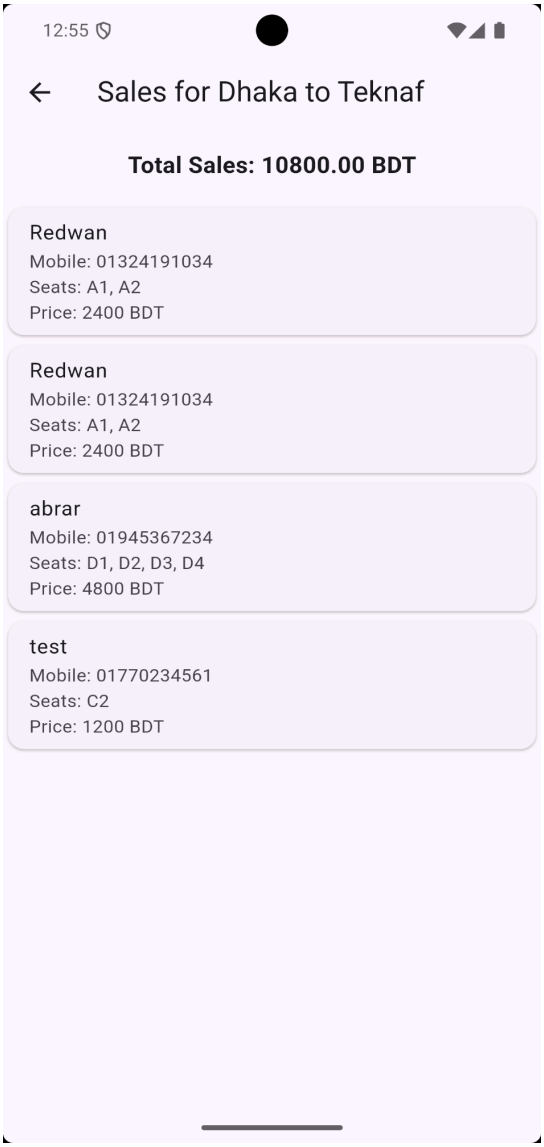



Figure12: Admin sales details

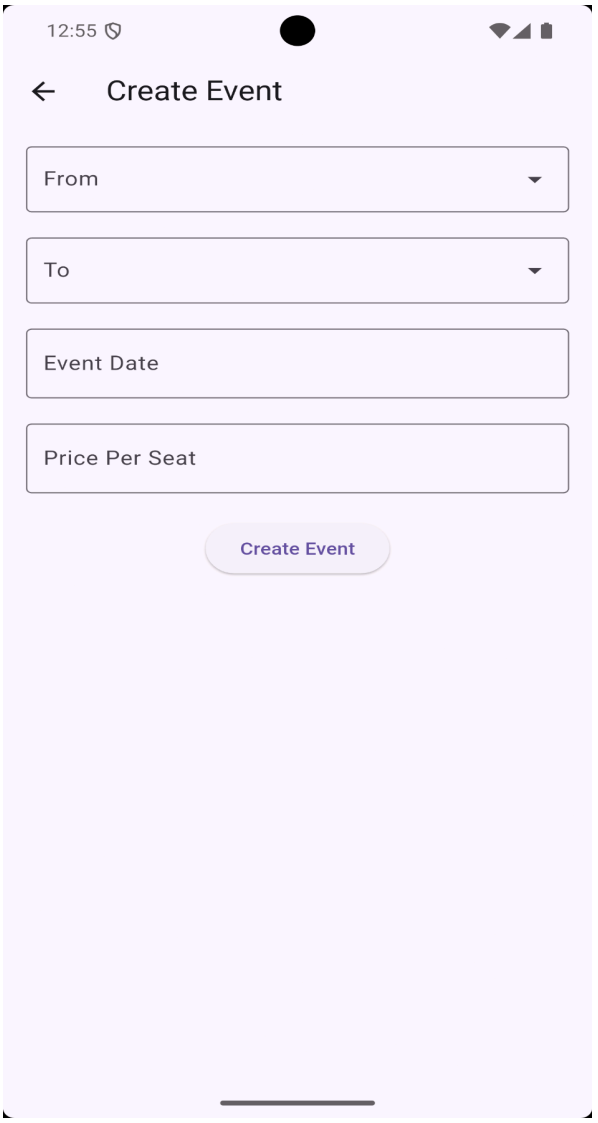



Figure13: Create events

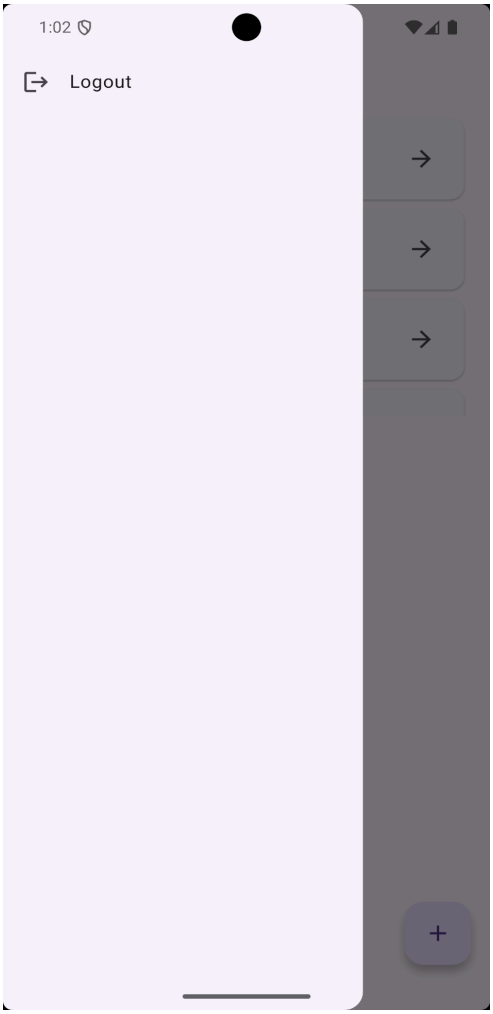



Figure14: Logout

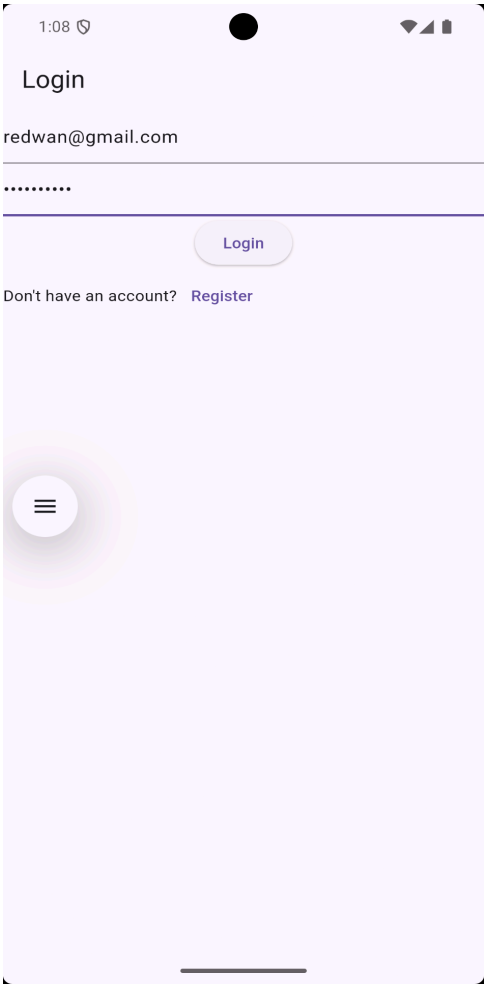



Figure15: Passenger login

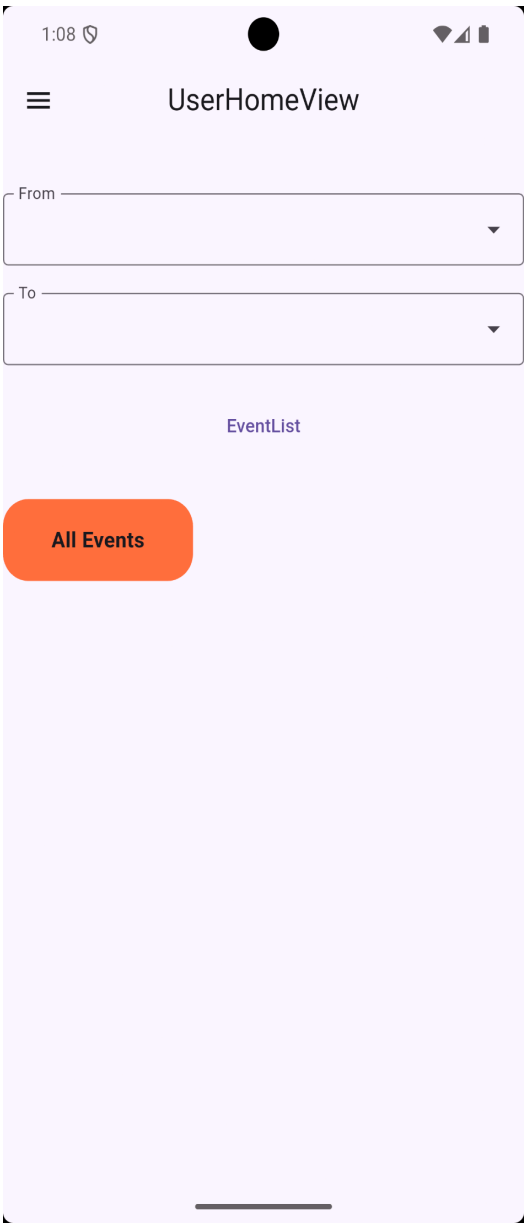



Figure16: User dashboard

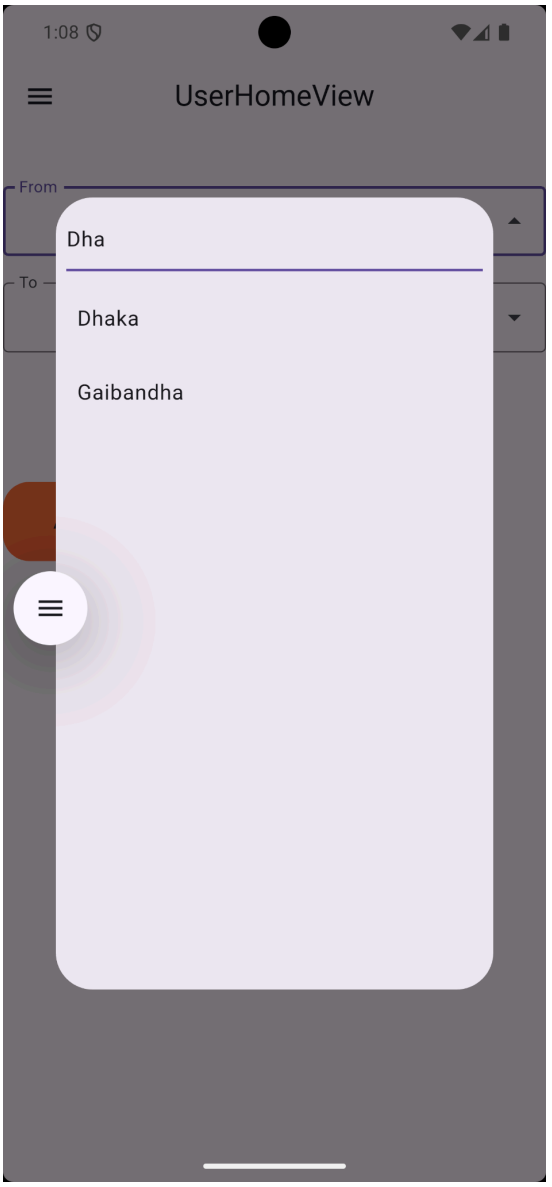
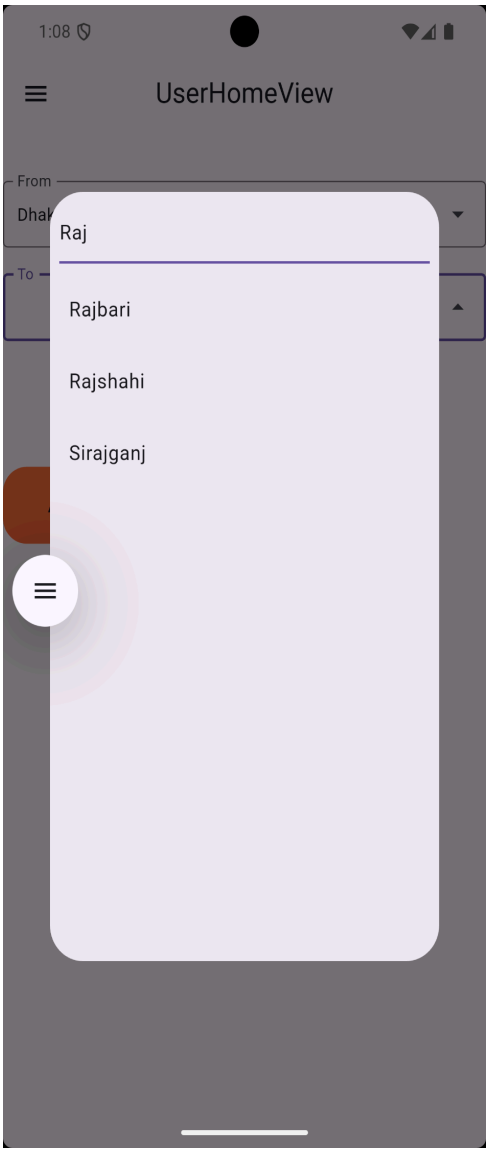



Figure17: Events search by location from



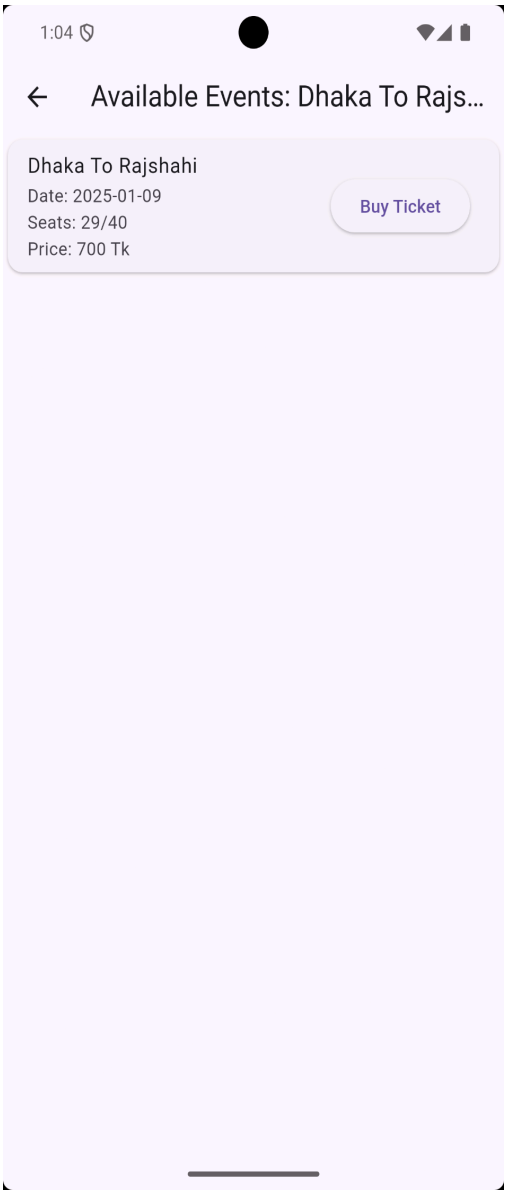




Figure18: Events search by location to

Figure19: Events Found

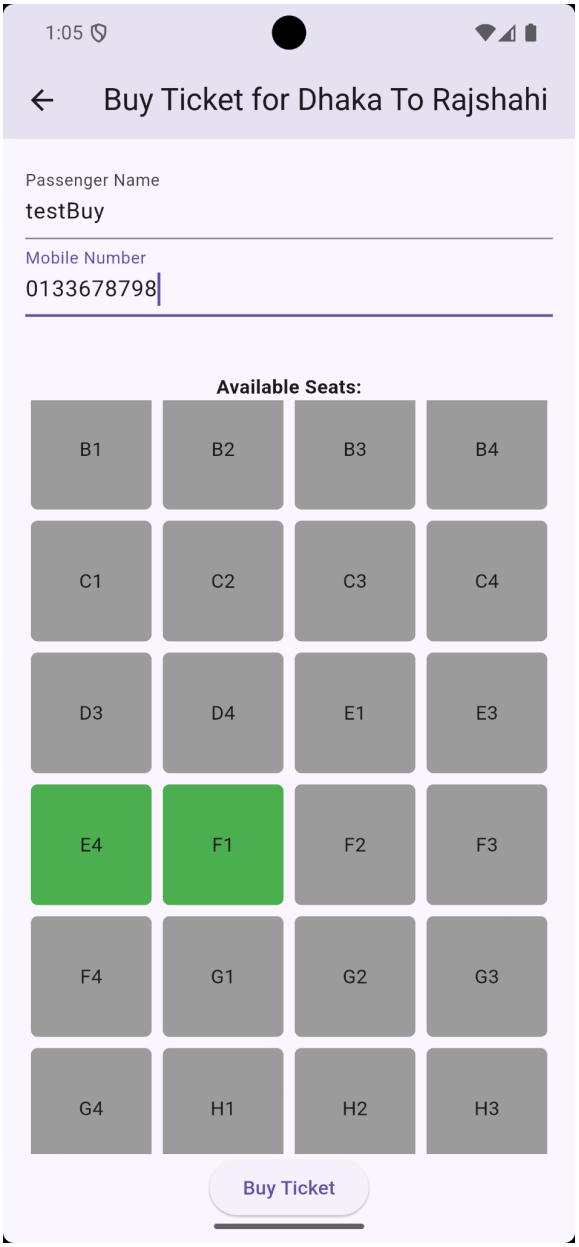



Figure20: Select sit


Figure21: Purchased Done

## 5.3 Summary

This chapter provided a comprehensive user manual for the Ticket Reservation System, ensuring that users can navigate and utilize its features efficiently. The manual included step-by-step instructions for signing up, logging in, booking tickets, managing profiles, and accessing offline functionalities. By simplifying complex processes and offering clear guidance, the user manual aims to enhance the user experience and ensure the system's functionality is accessible to all. This document serves as a critical resource for users to maximize their interaction with the application.

# Chapter 6 Project Summary

## 6.1 Introduction

This chapter provides a comprehensive summary of the Ticket Reservation System project, including its goals, development process, and key outcomes. It reflects on the challenges encountered during implementation, highlights the achievements, and evaluates how the project meets its intended objectives. By encapsulating the entire journey from inception to deployment, this chapter offers a consolidated view of the project's success and areas for future improvement.

## 6.2 Project Limitations

While the Ticket Reservation System achieved its primary objectives, certain limitations were encountered during the project development process. These include:

1. **Time Constraints**:
   The tight project schedule restricted the inclusion of advanced features, such as AI-based ticket recommendations and comprehensive predictive analytics for user behavior.
2. **Budgetary Restrictions**:
   Financial constraints limited access to premium tools and services, resulting in reliance on free or open-source technologies. This impacted the scope of the project and the ability to implement highly scalable infrastructure.
3. **Technical Challenges**:
   - Compatibility with older versions of Android and iOS was a challenge, restricting the application’s reach to users with outdated devices.
   - Real-time updates for seat availability and booking confirmations were constrained by limited server capabilities.
4. **Incomplete Features**:
   Certain planned features, such as multi-language support and enhanced accessibility options for users with disabilities, had to be deferred for future iterations due to resource and time limitations.
5. **Dependence on Internet Connectivity**:
   While the system provides offline access to previously booked tickets, core functionalities like ticket searching and new bookings require active internet connectivity, limiting usability in areas with poor network coverage.
6. **Testing Constraints**:
   Comprehensive testing, especially stress testing under peak user loads, was partially limited. This leaves some uncertainties regarding performance under high demand scenarios.

## 6.3 Scope

The Ticket Reservation System is designed to provide an efficient and user-friendly solution for booking tickets for events and transportation. The project focuses on delivering core functionalities that address the key needs of users while maintaining a balance between usability and performance. Below is an outline of the features and boundaries of the project:

**Features and Modules Covered**

1. **User Account Management**:
    - Signup and login functionality with third-party authentication (e.g., Google, Facebook).
    - Secure password recovery and multi-factor authentication.
2. **Ticket Booking System**:
    - Search for tickets by event, location, or date.
    - Seat selection (where applicable) and real-time availability checks.
    - Support for multiple payment methods, including credit cards and digital wallets.
    - Booking history and quick rebooking options.
3. **Notifications and Alerts**:
    - Real-time notifications for booking confirmations, reminders, and cancellations.
    - System alerts for payment issues or application updates.
4. **Offline Accessibility**:
    - Access to previously booked tickets without an active internet connection.
5. **Admin Panel**:
    - Management of events, ticket availability, and pricing.
    - User account management and customer support tools.
    - Generation of sales and user engagement reports.
6. **Profile Management**:
    - Edit personal details and update payment methods.
    - View and manage past bookings.

**Exclusions and Boundaries**

1. **Advanced AI Integration**:
    - Features like AI-based ticket recommendations or predictive analytics are not included due to time and budget constraints.
2. **Multi-Language Support**:
    - The system is currently available in a single language and does not include translations or multi-language capabilities.
3. **Accessibility Enhancements**:
    - Comprehensive features for users with disabilities, such as screen readers or voice commands, are excluded in this version.
4. **Scalability Beyond Initial Deployment**:

    - While the system supports up to 1,000 concurrent users, larger-scale deployment and associated infrastructure are beyond the scope of this phase.
5. **Integration with External Systems**:
    - The project does not include integration with external event platforms, third-party ticket distributors, or travel agencies.
6. **Custom Event Management**:
    - Users cannot create or manage their own events within the system.

## 6.4 Future Work

The Ticket Reservation System has laid a solid foundation for an efficient and user-friendly booking platform. However, several areas remain for future enhancements and development to further enrich the user experience and scalability of the application. These include:

**1. Advanced Features and Functionalities**

- **AI-Powered Recommendations**:
  Integrating machine learning algorithms to provide personalized ticket suggestions based on user preferences, booking history, and trends.
- **Dynamic Pricing Models**:
  Implementing dynamic pricing algorithms to adjust ticket costs based on demand, availability, and time of booking.
- **Real-Time Updates**:
  Enhancing the system to provide live updates for seat availability and event changes using robust real-time databases like Firebase Realtime Database.

**2. Accessibility Enhancements**

- **Multi-Language Support**:
  Expanding the application to support multiple languages to cater to a global audience.
- **Features for Users with Disabilities**:
  Adding support for screen readers, voice commands, and high-contrast themes to improve accessibility for users with disabilities.

**3. Expanded Integration**

- **Third-Party Event Platforms**:
  Integrating with popular event platforms (e.g., Eventbrite) to offer a wider range of tickets and events.

- **Payment Gateways**:
  Adding support for additional payment methods, including cryptocurrency and regional payment solutions.
- **Calendar and Social Media Integration**:
  Enabling users to sync their bookings with personal calendars or share event details on social media.

**4. Scalability and Performance**

- **Cloud Infrastructure**:
  Migrating to a scalable cloud infrastructure to support larger user bases and peak traffic loads.
- **Load Testing and Optimization**:
  Conducting comprehensive performance testing under high demand scenarios to ensure reliability and responsiveness.

**5. User Engagement**

- **Loyalty Programs**:
  Introducing reward points, discounts, and special offers to incentivize repeat bookings and user engagement.
- **Gamification**:
  Adding interactive features such as achievements or rewards for frequent usage to increase user interaction.

**6. Security Upgrades**

- **Enhanced Authentication**:
  Implementing biometric login options such as fingerprint or facial recognition for added security.
- **Compliance with Data Protection Laws**:
  Ensuring adherence to regional and international data privacy regulations, including GDPR and CCPA, for a global rollout.

**7. Event Creation and Management**

- **User-Generated Events**:
  Allowing users to create, manage, and sell tickets for their own events directly within the system.
- **Collaboration Tools**:
  Providing tools for event organizers to collaborate with team members within the platform.

**Further Research**

- **Behavioral Analytics**:
  Analyzing user behavior to understand preferences and improve system usability.

- **AR/VR Integration**:
  Exploring the use of augmented or virtual reality for event previews, seating arrangements, or virtual events.
- **Blockchain-Based Ticketing**:
  Investigating blockchain technology to prevent ticket fraud and improve transparency in transactions.

## 6.5 Conclusion

The Ticket Reservation System successfully achieved its objective of delivering a streamlined, user-friendly platform for booking tickets for events and transportation. Key achievements include the development of core features such as ticket booking, notifications, and offline access, as well as ensuring system security and reliability.

The project demonstrated the importance of understanding user needs and focusing on simplicity and functionality. Lessons learned include the necessity of rigorous time management, resource allocation, and continuous testing to overcome challenges such as budget constraints and technical complexities.

While certain limitations exist, such as incomplete features and scalability constraints, the system provides a strong foundation for future enhancements. The project aligns with its goals by offering a solution that improves efficiency, accessibility, and user satisfaction in the ticketing process, paving the way for further research and development.

# REFERENCES


[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[2] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.

[3] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 9th ed. McGraw-Hill Education, 2020.

[4] Firebase Documentation, [Online]. Available: https://firebase.google.com/docs. [Accessed: Dec. 25, 2024].

[5] Flutter Documentation, [Online]. Available: https://flutter.dev/docs. [Accessed: Dec. 25, 2024].

[6] SQLite Documentation, [Online]. Available: https://sqlite.org/docs.html. [Accessed: Dec. 25, 2024].

[7] W3C, "Web Content Accessibility Guidelines (WCAG)," [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/. [Accessed: Dec. 25, 2024].

[8] Google Play Console Help, [Online]. Available: https://support.google.com/googleplay/android-developer/. [Accessed: Dec. 25, 2024].

[9] ISO/IEC 25010:2011 Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models. International Organization for Standardization, 2011.

# APPENDICES

## Appendix B: Code Implementation

```dart
class BuyTicketPage extends StatefulWidget {
  final String eventId;
  final String eventName;
  final double pricePerSeat;
  final Map<String, dynamic> eventData;

  BuyTicketPage({required this.eventId, required this.eventName, required this.pricePerSeat, required this.eventData});

  @override
  _BuyTicketPageState createState() => _BuyTicketPageState();
}

class _BuyTicketPageState extends State<BuyTicketPage> {
  final TextEditingController nameController = TextEditingController();
  final TextEditingController mobileController = TextEditingController();
  List<String> selectedSeats = [];

  final PaymentService paymentService = PaymentService();

  @override
  Widget build(BuildContext context) {
    final availableSeats = widget.eventData['availableSeats'] as List<dynamic>;
    final pricePerSeat = widget.eventData['pricePerSeat'];
    return Scaffold(
      appBar: AppBar(
        title: Text('Buy Ticket for ${widget.eventName}'),
      ),  // AppBar
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: nameController,
              decoration: InputDecoration(labelText: 'Passenger Name'),
```

final_redwan.docx

ORIGINALITY REPORT

| 12% | 8% | 1% | 7% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | dspace.daffodilvarsity.edu.bd:8080<br>Internet Source | 3% |
| 2 | 1library.net<br>Internet Source | 1% |
| 3 | Submitted to Higher Education Commission Pakistan<br>Student Paper | 1% |
| 4 | Submitted to University of Greenwich<br>Student Paper | 1% |
| 5 | Submitted to Macquarie University<br>Student Paper | <1% |
| 6 | Submitted to University of Westminster<br>Student Paper | <1% |
| 7 | Submitted to KDU College Sdn Bhd<br>Student Paper | <1% |
| 8 | Submitted to University of Technology, Sydney<br>Student Paper | <1% |
| 9 | Submitted to London Churchill College | |

turnitin

# Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

| | |
|---|---|
| Submission author: | Md. Redwanul Haque 172-35-2116 |
| Assignment title: | Check2 |
| Submission title: | final_redwan.docx |
| File name: | final_redwan.docx |
| File size: | 2.1M |
| Page count: | 90 |
| Word count: | 10,549 |
| Character count: | 64,191 |
| Submission date: | 04-Jan-2025 03:14PM (UTC+0600) |
| Submission ID: | 2559699372 |

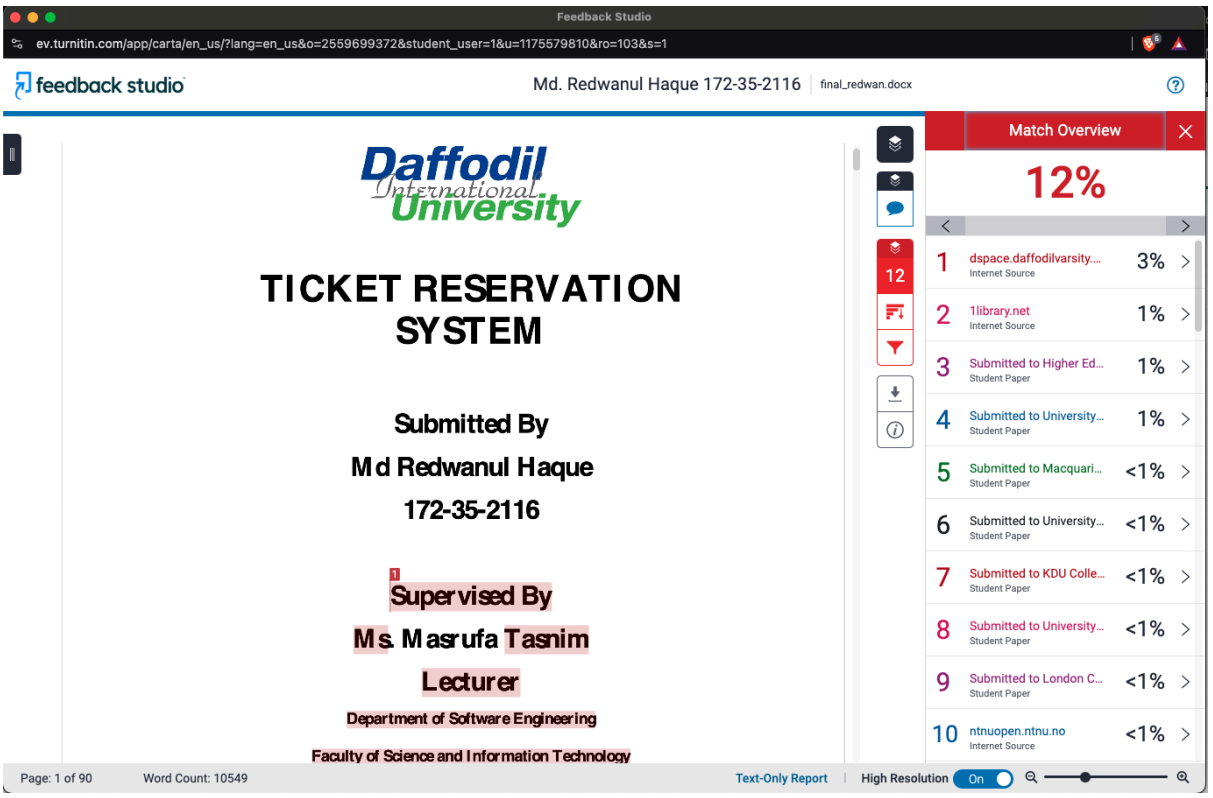
Daffodil International University

**TICKET RESERVATION SYSTEM**

**Submitted By**
**Md Redwanul Haque**
**172-35-2116**

**Supervised By**
**Ms. Masrufa Tasnim**
**Lecturer**
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

This project report has been submitted in fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering


Feedback Studio
ev.turnitin.com/app/carta/en_us/?lang=en_us&o=2559699372&student_user=1&u=1175579810&ro=103&s=1
feedback studio
Md. Redwanul Haque 172-35-2116
final_redwan.docx
Daffodil International University
TICKET RESERVATION SYSTEM
Submitted By
Md Redwanul Haque
172-35-2116
Supervised By
Ms. Masrufa Tasnim
Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Match Overview
12%
1 dspace.daffodilvarsity.... 3%
2 1library.net 1%
3 Submitted to Higher Ed... 1%
4 Submitted to University... 1%
5 Submitted to Macquari... <1%
6 Submitted to University... <1%
7 Submitted to KDU Colle... <1%
8 Submitted to University... <1%
9 Submitted to London C... <1%
10 ntnuopen.ntnu.no <1%
Page: 1 of 90
Word Count: 10549
Text-Only Report
High Resolution
On